Graduate School ETD Form 9
(Revised 12/07)

PURDUE UNIVERSITY
GRADUATE SCHOOL
Thesis/Dissertation Acceptance

This is to certify that the thesis/dissertation prepared

By Ryan Thomas Rybarczyk

Entitled
e-DTS 2.0: A Next-Generation of a Distributed Tracking System

For the degree of Master of Science

Is approved by the final examining committee:

Dr. Rajeev Raje
Chair

Dr. Mihran Tuceryan

Dr. Panos Linos

To the best of my knowledge and as understood by the student in the *Research Integrity and Copyright Disclaimer (Graduate School Form 20)*, this thesis/dissertation adheres to the provisions of Purdue University's "Policy on Integrity in Research" and the use of copyrighted material.

Approved by Major Professor(s): Dr. Rajeev Raje

Approved by: Dr. Rajeev Raje 11/17/2010
Head of the Graduate Program Date

Graduate School Form 20
(Revised 9/10)

PURDUE UNIVERSITY
GRADUATE SCHOOL

Research Integrity and Copyright Disclaimer

Title of Thesis/Dissertation:

e-DTS 2.0: A Next-Generation of a Distributed Tracking System

For the degree of Master of Science

I certify that in the preparation of this thesis, I have observed the provisions of *Purdue University Executive Memorandum No. C-22,* September 6, 1991, *Policy on Integrity in Research.**

Further, I certify that this work is free of plagiarism and all materials appearing in this thesis/dissertation have been properly quoted and attributed.

I certify that all copyrighted material incorporated into this thesis/dissertation is in compliance with the United States' copyright law and that I have received written permission from the copyright owners for my use of their work, which is beyond the scope of the law. I agree to indemnify and save harmless Purdue University from any and all claims that may be asserted or that may arise from any copyright violation.

Ryan Thomas Rybarczyk

Printed Name and Signature of Candidate

11/11/2010

Date (month/day/year)

*Located at http://www.purdue.edu/policies/pages/teach_res_outreach/c_22.html

E-DTS 2.0: A NEXT-GENERATION OF A DISTRIBUTED TRACKING SYSTEM

A Thesis

Submitted to the Faculty

of

Purdue University

by

Ryan Thomas Rybarczyk

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science

December 2010

Purdue University

Indianapolis, Indiana

To my parents, my sisters, and my wife

ACKNOWLEDGMENTS

I would like to start off by thanking my professor, advisor, and mentor, Dr. Rajeev Raje, for giving me the opportunity to be research assistant under his guidance. His support and guidance has allowed me to grow and become a much more capable researcher. I am forever grateful for his support and encouragement through the process of preparing this thesis. I would also like to thank my co-advisor, Dr. Mihran Tuceryan, as his guidance and knowledge of augmented reality allowed him to provide helpful insight into many problems that I encountered throughout my study. Finally, I would like to Dr. Panos Linos for agreeing to serve as my third member of my graduate committee. While his role was not large he provided support and guidance with his helpful knowledge and insight into Software Engineering concepts and practices.

I would like to thank my family, most notably my parents – for without their everlasting love and support I would not have been able to achieve this goal in my life. I would also like to thank my wife, Amy, for her support, guidance, and patience throughout the entire process. Often times my work consumed my time but she was always there for me – and for that I forever grateful.

Finally, I would like to thank the entire Computer Science Department at IUPUI. They helped to provide an environment where I could strive and provide positive contributions to the field of Computer Science; all the while gaining an invaluable education. In addition to this they allowed me to server many roles including teaching assistant, tutor, and instructor which allowed me to grow as a teacher and an individual.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

DTS	Distributed Tracking System
e-DTS	Enhanced Distributed Tracking System
EEDT	End-To-End Discovery Time
EERT	End-To-End Response Time
SRG	Spatial Relation Graph

ABSTRACT

Rybarczyk, Ryan Thomas. M.S., Purdue University, December, 2010. e-DTS 2.0: A Next-Generation of a Distributed Tracking System. Major Professor: Rajeev Raje.

A key component in tracking is identifying relevant data and combining the data in an effort to provide an accurate estimate of both the location and the orientation of an object marker as it moves through an environment. This thesis proposes an enhancement to an existing tracking system, the enhanced distributed tracking system (e-DTS), in the form of the e-DTS 2.0 and provides an empirical analysis of these enhancements. The thesis also provides suggestions on future enhancements and improvements. When a Camera identifies an object within its frame of view, it communicates with a JINI-based service in an effort to expose this information to any client who wishes to consume it. This aforementioned communication utilizes the JINI Multicast Lookup Protocol to provide the means for a dynamic discovery of any sensors as they are added or removed from the environment during the tracking process. The client can then retrieve this information from the service and perform a fusion technique in an effort to provide an estimation of the marker's current location with respect to a given coordinate system. The coordinate system handoff and transformation is a key component of the e-DTS 2.0 tracking process as it improves the agility of the system.

CHAPTER 1 INTRODUCTION

There are many applications, in fields such as health informatics, in which there is a need and desire to track an object as it moves through time and space. In vision-based tracking, devices such as optical cameras are interfaced with software in order to provide an estimate of an object's location or pose. These devices can then be grouped together and connected via a Local Area Network (LAN) to form a federation of tracking devices. When these devices are grouped together, they form a distributed system of tracking sensors which can communicate in an effort to provide tracking estimates. The idea of a federation of tracking devices allows for broader tracking of an object as it moves through a space. The results of the various tracking devices can then be combined together in an effort to provide a more accurate estimate as to the location of an object; this process is known as data fusion [2]. A well-defined and agile tracking framework is desired to serve as the basis for tracking research and discovery.

This thesis proposes enhancements to an existing distributed tracking system, the Enhanced Distributed Tracking System (e-DTS) proposed in [2], in the form of the Enhanced Distributed Tracking System 2.0 (e-DTS 2.0) and provides an empirical evaluation of the proposed system. The concept of the e-DTS is that many inexpensive sensors, such as Web cameras, can be used to create a comprehensive and efficient distributed tracking infrastructure for indoor tracking. This design was demonstrated to be scalable and included a Kalman-based fusion technique, as shown in [2]. In the e-DTS, there is no concept of a shared global coordinate system or a coordinate handoff. This prevents the cameras from dynamically discovering the tracking environment they are in and establishing a single coordinate system to use when estimating the position of

the tracking marker. Because of this, the coordinate system must be established prior to execution of tracking and each camera must be manually calibrated with these coordinates. The e-DTS also does not handle network faults or interruptions gracefully, as a halt or abnormal termination of one service or the JINI Lookup Service would prevent further tracking of the system. The e-DTS 2.0 proposed in this thesis attempts to solve all of these issues.

1.1 Need for Dynamic Discovery in a DTS

Tracking systems rely on sensors to describe their surrounding environment as well as the relative location of an object as they know it within that environment. In visual tracking systems, cameras play the role of sensors within the system. As the environment and tracking system setup evolve, i.e., cameras are added or removed, the system must be able to remain dynamic in an effort to provide reliable tracking of an object. As an object travels through an environment, it will come in and out of being tracked by any particular tracking sensor. This phenomenon in which the object is blocked from the view of the camera is known as occlusion. This means that the camera is not able to physically recognize or see the object during this period of obstruction [33]. Because of this obstruction, multiple cameras can be utilized in the hope that one of the cameras may always be seeing the object as it moves through the environment in order to provide a full coverage. A multi-camera environment in which many different cameras are working together to provide an accurate estimate this issue is far less severe or prevalent. Because of this distributed setup of cameras, there is a need to dynamically locate and identify other cameras within the distributed tracking setup. Because of failures by the underlying network, services, or the cameras themselves there is the need for the system to adapt and handle change by the additional and removal of various components within the system.

As an object moves through an environment, different cameras within that environment will come into view and begin seeing the object. When this seeing occurs, the camera needs to be able to communicate this information to the client in an effort to provide an accurate tracking estimate as to the current location of the object. This camera communication is done over a local area network (LAN) and allows for the sharing and propagation of tracking information. Because both sensors and the underlying networks can be unreliable, a dynamic means for locating available resources used in tracking is essential.

In a static environment, each tracking service and its corresponding Camera is aware of the location of the central service that is then used to provide the tracking information to the user. In a static environment no discovery is needed as each service is explicitly made aware of the location of this central service during the setup. This design is not agile for real world applications, as it limits the tracking system in its ability to provide accurate tracking of an object in a dynamic and ever changing environment. For instance, if the central service, mentioned above, fails or is unable to be reached then the entire system will either fail or be rendered useless as no information can be passed or provided to the user. This occurs because of the predetermined and static network topology established by the tracking system upon startup. In addition to this, cameras may come or go within the network – in a static environment this is not possible as the network of cameras must remain stable and cannot handle random changes. This need for the dynamic discovery and communication will allow for the camera services to also gain the important factor of being more fault-tolerant than with the previous static discovery used .

1.2 Need for a Coordinate System Handoff in a DTS

In a distributed tracking system, there is the desire to provide an accurate estimate of the physical location of an object within a Three Dimensional (3D) world. In order to achieve this goal, there must be a well-established point of

origin that is known or can be provided to the cameras to use in their calculation of the location of the object. This can be achieved with the concept of a Global Coordinate System in which each camera is calibrated with respect to its origin. Each camera is calibrated with respect to the origin and then it uses this information when determining the estimated location of the object in terms of the Global Coordinate System.

As an object moves from one known environment to another, a coordinate system handoff is needed so that the tracking information transition and estimation can be provided to the user. This handoff involves the transition from one coordinate system to another. Therefore, with the ability for a shared coordinate system and coordinate system handoff, there is an opportunity to track an object as it moves between multiple tracking environments and to be able to correctly and seamlessly provide a tracking estimate for an object during this transition.

This is a limitation of both the DTS and the e-DTS as proposed by [1] and [2] respectively. Neither of the two proposed systems implemented or utilized a global coordinate system using coordinate handoff and transformation. This limited the tracking of an object to only one known coordinate system. Therefore there is a need for an enhancement to these systems in an effort to provide dynamic tracking of objects.

1.3 Need for Improved Calibration in a DTS

In vision-based tracking systems, a camera is the primary means of tracking an object. Therefore, the most important activity that must be done with any camera within the system is calibration. Because of this, there is a need for an accurate and efficient method of calibrating the camera. There are many tools that allow such a calibration but most are not intended for calibration or accuracy on a scale for wide area tracking. Calibration is a two-step process that first involves preparing the camera by gathering its extrinsic and intrinsic parameters

[2]. The second step of camera calibration is to generate a file that can be used by the camera during the tracking process. Therefore, improved calibration of the cameras will allow for more accurate estimations provided during the tracking process, and thus helping to improve the overall tracking system.

1.4 Need for Clock Synchronization in a DTS

In any distributed system, one of the key requirements is the clock synchronization between two or more communicating devices. The clock synchronization is needed as machine clocks over time will drift thus causing inconsistencies between various clocks within the system. This drift is a major factor when attempting to fuse data results together to provide accurate tracking. This drift will cause fused results to compute any inaccurate estimation of the location of an object or due to clock differences will prevent fusion altogether. Therefore, with improved clock synchronization better and more accurate fusion can take place within the tracking system.

1.5 Need for Quality of Service Parameters in a DTS

In a wide scale tracking application sensors may vary in terms of the quality of service (QoS) that they are able to provide during the tracking process. In addition, the QoS may drive the selection of a subset of sensors in the tracking system. The ability to select cameras and their data based upon their QoS parameters will lead to the ability to provide higher QoS when it comes to tracking of an object. Therefore, there is a need to indicate the QoS parameters for each of the cameras to allow for a proper selection of the cameras involved in the tracking system.

1.6 Issues to be Resolved

The above listed needs highlight the importance in making improvements in the field of distributing tracking systems. Therefore, this thesis will attempt to tackle a selection of these issues.

In a dynamic and changing environment in which objects and cameras may be added or removed teaming up with the unreliability of the underlying communication network there is a need for the e-DTS to provide dynamic discovery and communication between participating devices. This would eliminate a central point of failure and provide the ability for the addition or removal of cameras to happen seamlessly during the tracking process. This dynamic discovery and communication would also allow for increased knowledge of the topology of the tracking system as known by the cameras as well as the tracker client.

In the tracking of an object between two different coordinate systems there is a transition point in which one coordinate system must be transformed into the other coordinate system. For this transformation to take place a mechanism and global coordinate system must be established and put in place. This creates a need in the e-DTS as no coordinate system handoff or global coordinate system are utilized during the tracking process. Therefore additions can be made such that the e-DTS is able to properly handle coordinate system transformation and the subsequent handoff from one coordinate system to another during the tracking process.

The fundamental key aspect of any tracking application is the accuracy of the estimated calculation of an object during any moment in time. In vision based tracking systems this is typically dominated by the type and quality of camera and the subsequent calibration of the camera. Because of the goal and mission of the e-DTS to provide tracking using inexpensive tracking devices, in the form of web cameras, the calibration is critical to the overall tracking accuracy of the system. Therefore alternative methods, from that provided with ARToolkit, and

techniques to camera calibration could yield better camera calibration and in turn yield more accurate tracking results.

In any distributed tracking system clock synchronization is the key for maintaining accurate communication between hosts. During the fusion process in distributed tracking the results of various cameras must be combined only when their timestamp readings are the same to insure accurate tracking. Therefore there is need for the e-DTS to incorporate clock synchronization methods or tools in an effort to improve the overall accuracy and the ability to fuse tracking data by the cameras.

Quality of Service (QoS) parameters are useful in identifying items based upon select attributes that they provide. In vision based tracking systems QoS parameters may be found within the cameras, the software, or the hardware provided with the system. In the e-DTS there is a need to provide the ability to select cameras and their services based upon QoS parameters that they may provide.

1.7 Problem Statement

The e-DTS, as proposed in [2], utilizes the Unicast protocol as a means for service registration and discovery. The e-DTS 2.0 is proposing the implementation and usage of the Multicast protocol as a means for service discovery and registration. This will allow for dynamic discovery and registration of the services. In the e-DTS there was no acceptable approach utilized to handle clock drift between the various devices present, and so therefore fusion activities were negatively impacted. This thesis proposes potential solutions to the issue and problem of clock drift between the various devices in a distributed setup. This proposed solution will allow for time constrained fusion to take place within the e-DTS 2.0. The e-DTS utilized the basic camera calibration method as described in [3]. This thesis will provide an analysis of alternative methods of camera calibration in an effort to improve tracking estimates. In the e-DTS there

was no concept or implementation of a global coordinate system or any presence of a coordinate handoff between worlds when tracking an object. The e-DTS 2.0 will implement a global coordinate system and will also attempt to demonstrate a solution to coordinate system handoff when tracking of an object through multiple worlds. Finally, the e-DTS 2.0 will introduce the concept of QoS parameters during the tracking process for selection of camera services and their data.

1.8 Hypothesis

The goal of this thesis is to show that tracking of an object through an environment can be achieved by using inexpensive vision-based tracking sensors in a dynamic and agile fashion.

1.9 Contributions

This thesis will build upon and enhance the existing features and functionality of the e-DTS as described in [2] through the creation of the e-DTS 2.0. Contributions include demonstrating the ability of the e-DTS 2.0 to handle and utilize dynamic network discovery. It will demonstrate the higher level of fault tolerance of the e-DTS 2.0 over the e-DTS by the addition and removal of camera and lookup services. It will provide empirical analysis and results that demonstrate that improved fusion can take place with improved clock synchronization through the usage of software tools. It will provide analysis on various techniques for camera calibration in tracking systems. It will demonstrate the usefulness and ability for the e-DTS 2.0 to utilize QoS parameters in camera selection and tracking of an object. Finally, this thesis will demonstrate the ability of a distributed tracking system to be able to handle and utilize a shared and global coordinate system and provide coordinate system handoff and transformation as an object moves from one coordinate system to another.

1.10 Organization

This thesis is organized into five chapters. The first of these chapters includes the introduction, the need for an e-DTS 2.0, the existing problems and proposed solutions along with the goals of this thesis. The second chapter includes a review of related work to that of this thesis. The third chapter presents an outline of the e-DTS as well as the proposed modifications and enhancements that are contained with this thesis' goals. The fourth chapter presents the results from experimentation and a discussion of these results and how they relate to the problems and goals discussed in chapter one. The fifth and final chapter presents a conclusion of the work related to this thesis as well as some suggested areas of future work or extension of the e-DTS 2.0.

CHAPTER 2 RELATED WORK

There are many applications for which tracking of an object as it moves throughout an environment is necessary. With this demand for such applications there come many such problems facing distributed tracking. These problems have been identified and studied by researchers in many different disciplines.

One fundamental goal of tracking systems is the desire to estimate the location and orientation of an object in real-time. As an object moves through an environment it may move too quickly for any tracking sensors to be able to accurately identify it and locate it in the scene. This prevents the object from being tracked and tracking information from being gathered and displayed to the user. Another potential problem within this realm of real-time tracking is the possibility that while the tracking sensor may be able to locate the object and provide tracking information the object may still be moving too quickly for any calculations or fusion to be performed on the data, thus leaving the tracking information supplied to the user stale and no longer relevant. The area of real-time tracking has been subject to many studies in an effort to improve upon the ability to track an object in real-time.

The intended use of any tracking system is the key when determining the QoS attributes that the system must meet. This knowledge will drive both the accuracy of the estimated position to the timing constraints on the retrieval of information from the tracker. For instance, a system that is tracking the location of equipment within a room will have a different set of requirements and parameters than that of a person moving through a room. In each instance both timing and accuracy are important but each application will require its own requirements. In addition to this a discussion of the environment in which the

tracking system will be deployed is critical. This will drive the type of tracking sensors being used as well as the means for communication. This requirement is based on a desire to provide a flexible system that can evolve and handle faults over time. Once a design has been developed, the implementation of the tracking sensors must take place.

Many tracking systems have been developed or worked in conjunction with Augmented Reality (AR) applications. The reason for this trend is the fact that AR applications must be able to properly identify a pattern or marker that can then provide the basis for overlaying an image on top of it to form an augmented scene. Because of this nature of the application the location of the marker must be able to be identified and utilized in a real-time manner.

Another major application that utilizes the tracking domain is that of robotics. In robotics vision-based tracking is key in identifying the scene and estimating the position for the robots movements. In this domain the ability to identify the environment and accurate estimate location in a real-time manner is required. Because of this a need of high accuracy is required as poor calibration leading to poor estimation could lead to catastrophic results for the robot.

2.1 Related Work in Camera Calibration

In vision based tracking systems quality camera calibration is key. Because of this, there are various techniques and methods available that provide camera calibration. In most camera based calibration techniques and methods calibration takes place using physical measurement of the various points in 3D. The calibration of the camera is based on the corresponding 2D image points of these 3D points and using the projective geometry constraints to estimate the camera parameters. The camera parameters are usually divided into intrinsic parameters (focal length, principal point) and extrinsic parameters (pose of the camera relative to the reference coordinate system).

In [3], Kato and Billinghurst utilize the ARToolkit API to conduct an experiment using Head Mounted Displays [HMD] as their method of vision based tracking to demonstrate marker tracking and calibration. Their work also applies to camera based tracking systems as the primary goal is the estimation of the pose of the camera. They state that one of the primary shortcomings of the ARToolkit API's calibration method is that of correctly recognizing the calibration pattern and properly estimating the pose. Because of this shortcoming the overall accuracy of the calibration will be negatively impacted as well as the overall tracking estimation provided. Thus, their work shows the importance of camera calibration as it relates to accuracy estimates when using the ARToolkit API.

A later study done in [13] highlights the overall importance of camera calibration when using the ARToolkit API. Malbezin et al. designed an experiment to examine the accuracy of calibration at varying distances in a static environment. In this work they demonstrated that the error of the calibration increases as the physical distance between the calibration pattern and the camera itself increases. They also show that the accuracy of the calibration is affected by the angle of the camera with respect to that of the calibration pattern. They make a suggestion that correction filters for detecting the angle and distance of the camera from the calibration pattern could help resolve some of the accuracy issues that they discovered. As a result, camera calibration when using the ARToolkit API should be done within close proximity (i.e. 1 meter) to the physical location of the camera and on the same relative plane.

In [14], Claus and Fitzgibbon discuss the issues of real-world vision based tracking scenarios and how they differ from the stable laboratory conditions. As noted in [3] the lighting as well as interference from other objects can hinder the camera calibration process. They attempt to suggest utilizing patterns that are easy to recognize under less than ideal conditions. This would aid both the calibration and tracking of an object in a real-world scenario and environment. In a subsequent study [15], they suggest using an alternate technique for camera calibration, the use of a structure-from-motion (SFM) solver. This technique uses

information captured from camera motion and scene structure from a series of images captured by the device in an effort to provide an estimation of pose. Their work demonstrates how this technique can affect the overall tracking accuracy of an object as it moves through time and space within an environment.

The above listed work emphasizes the importance of camera calibration with relation to tracking accuracy in vision based tracking systems. Along with camera calibration, the selection and type of device, i.e. camera or head mounted display, is critical when judging accuracy as the higher quality the device the better calibration will take place and thus better and more accurate tracking.

2.2 Related Work in Dynamic Discovery

In distributed tracking there is a desire to be able to provide dynamic and agile tracking by the various devices participating within the system. This desire is fueled by the usage of real-world tracking applications that may or may not be aware of their environment and location and must discover their surroundings. Because the devices in a tracking system must communicate via a network protocol there is a need to be able to dynamically discover and communicate amongst the various devices or nodes of the system.

In a study done in [17], Baroody et al. examine the scalability of dynamic discovery using JINI in a vehicle network. They propose utilizing a dynamic discovery service (SDS) based JINI system where the various nodes in the ad-hoc network can communicate and discover one another. They evaluate their proposed system on the basis of scalability of the lookup service that is provided. This work relates directly to a distributed tracking vision based system in that the various cameras participating are essentially nodes that are active in an ad-hoc network setup. These nodes are responsible for discovering and sharing their knowledge with one another in the hopes of providing complete coverage of their environment. They show that such a system does in fact scale based upon the

number of lookups performed along with the number of nodes and network traffic within the system.

In [18], Chen et al. take a closer look at dynamic service discovery using alternatives to JINI. They propose a system in which they experiment with JINI and take a service-oriented approach to discovery and communication in a wireless ad-hoc network. Their proposed system is designed to utilize a hybrid discovery architecture in which they utilize both service-oriented and agent-oriented architectures. This work shows the limitations and abilities of the JINI architecture and its uses in real-world ad-hoc network discovery and communication.

Banaei-Kashani et al. present an interesting look on using peer-to-peer (p2p) discovery using web services in [25]. They introduce the concept of utilizing a fully decentralized discovery service approach using semantic-level matching. By using a p2p approach they allow the nodes in the network to handle the load and allow for the ability of node selection based upon self-contained attributes. In vision based tracking the cameras are the nodes of the system and can communicate in a p2p nature by using various self-contained attributes as methods for selection and discovery.

The above related work demonstrates the need and desire to provide agile discovery and communication within distributed tracking systems. Through the usage of technologies such as JINI this has been shown to be possible in a scalable and efficient manner. However, other trends have shown that alternative discovery and communication protocols must be used, the suggestion that decentralized p2p services could provide higher quality of service for communication amongst nodes in an ad-hoc network.

2.3 Related Work in Clock Synchronization

Extensive studies have been done in the area of clock synchronization and drift within distributed systems. Clock drift is the natural phenomena that

over time clocks will slowly drift apart from one another due to their chemical makeup. This drift negatively impacts tracking as the clocks of various machines may drift at different and variable rates causing imperfections and errors in estimated tracking accuracy and pose.

This work is headlined by that done by Lamport and Melliar-Smith in [20]. In this very important work they discuss the ability and concept of attempting to synchronizing clocks in the presence of faults. In real-time tracking systems the ability to have information collected at regular and identical times from various sensors is vital. They suggest the concept of a logical clock as a means for combating this clock drift found in machines. This concept removes the dependency on a physical clock and instead relies on the individual messages passed amongst the various machines in the distributed system as the concept of a clock.

With a dependency on a physical clock value due to real-time constraints, the concept proposed in [20] can still be utilized during the message passing and communication between the various devices and sensors in a distributed tracking system.

2.4 Related Work in Coordinate Systems and Spatial Relation Graphs

The need to coordinate transformation and handoff is a vital ability for a distributed tracking system to provide. This handoff and transformation has been studied and the suggestion that using spatial relation graphs (SRG) will allow for this handoff and transformation to take place. The following related work demonstrates and discusses this concept and its implementation in tracking systems.

In [16], Nagpal et al. demonstrate the ability to achieve accurate estimations of position and tracking in an ad hoc sensor network using a global coordinate system. Their goal was to demonstrate the ability for low power sensor networks to be able to discover their environment and accurately estimate

their position within a global coordinate system. They suggest that the nodes in the ad hoc network must first be able to form a group of nodes that can communicate to one another in an attempt to map the known environment. Once this has been achieved an estimation of the tracking position can be done through participating nodes. Their algorithm that they propose shows that an ad hoc network can produce a global coordinate system through self-organization.

In [21], Echtler et al. proposes using spatial relationship graphs (SRG) in Augmented Reality (AR) applications. They discuss how SRG's are more appropriately suited for distributed tracking using AR type applications as opposed to scene graphs. Their argument is that SRG's provide more flexibility and are more accurately mapped to that of real world physical coordinates than that of scene graphs. They split the scene graph into sub trees that are then encapsulated by the SRG and can then create an entire view of an environment. They built upon earlier work found in [28], in which Putska et al. suggested using spatial relationships for tracking and calibration in distributed tracking. They propose an algorithm for identifying and showing it to be possible to automatically construct a view of the network. Their focus on tracking and tracking accuracy using SRG's provides a way to establish patterns and apply self-learn tracking techniques.

This thesis attempts to address many of these related problems and to improve upon the work that has already been done in the field of distributed tracking. It will demonstrate the ability to implement a dynamic and flexible discovery system that will allow for fault-tolerance of the entire system. It will also tackle the issue of clock drift within distributed systems as to how it relates to a tracking system. Finally, it will attempt to demonstrate the ability to use a global coordinate system include the ability to transition between worlds while achieving an accurate estimate of the objects location.

2.5 Related Work in Tracking Systems

Many distributed tracking systems have been proposed, developed, and experimented on in an effort to help further tracking of objects through time and space. This work bridges many fields and disciplines all in the effort of furthering tracking accuracy by visual and sensor based tracking systems.

In [27], Jimenez proposes and studies a tracking system used in the field of health informatics. The Information Technology for Assisted Living at Home (ITALH) project is a camera-based tracking system to monitor elderly people who live without the need for a nurse. In this tracking system occlusion, or the blocking of the object from the cameras view, is one of the primary problems during the tracking process. The primary focus of this work is resolving the issue of occlusion within the tracking system and devising algorithms to handle that situation. As a result, this study classifies various types of occlusion that may occur within a camera-based tracking system and provides a geometrical approach to solving the issue of occlusion.

Soto et al., attempt to look at tracking various marker objects within a distributed tracking system in [34]. They propose using a Kalman-based technique that utilizes various neighboring cameras to form a consensus as to the actual physical state of the marker object. This process that they propose using is called a Kalman Consensus filter. They suggest that by using this technique the cameras within the system become self-aware and self-organizing and thus the cameras are able to learn the network topology over the tracking process. They are able to demonstrate the ability to track multiple objects over a wide area using a dynamic camera network.

In [1] and [2] a distributed tracking system using Cameras is proposed and experimented on. In [1] the system proposed utilizes both physical and virtual cameras while implementing a simple averaging fusion technique in an effort to provide an accurate tracking estimate. This work demonstrated that such a system could be created and used to track markers as they moved within a known environment. This system was then enhanced by [2] in which a Kalman-

based fusion technique was introduced as well as attributes to aid camera selection. In [2] a study was also conducted in an effort to show the scalability of the system with regards to both the fusion activity and the number of camera located with the tracking system. In addition to this, a calibration technique was introduced in an effort to improve the overall tracking accuracy. These two systems have provided the basis for the work done in this thesis.

The systems and work discussed above provides a good understanding of existing systems in the area of distributed tracking and outline the importance of creating dynamic and agile tracking systems using vision based sensors, or cameras.

CHAPTER 3 DESIGN AND IMPLEMENTATION

The previous chapter provided a background of related work in the area of Distributed Tracking. This chapter will take a closer look at the architecture and design behind the e-DTS [2]. Finally, it will provide a discussion on the proposed architecture and enhancements made in the creation of the e-DTS 2.0.

3.1 e-DTS Architecture

The Figure 3.1, shows the general architecture of the e-DTS system.

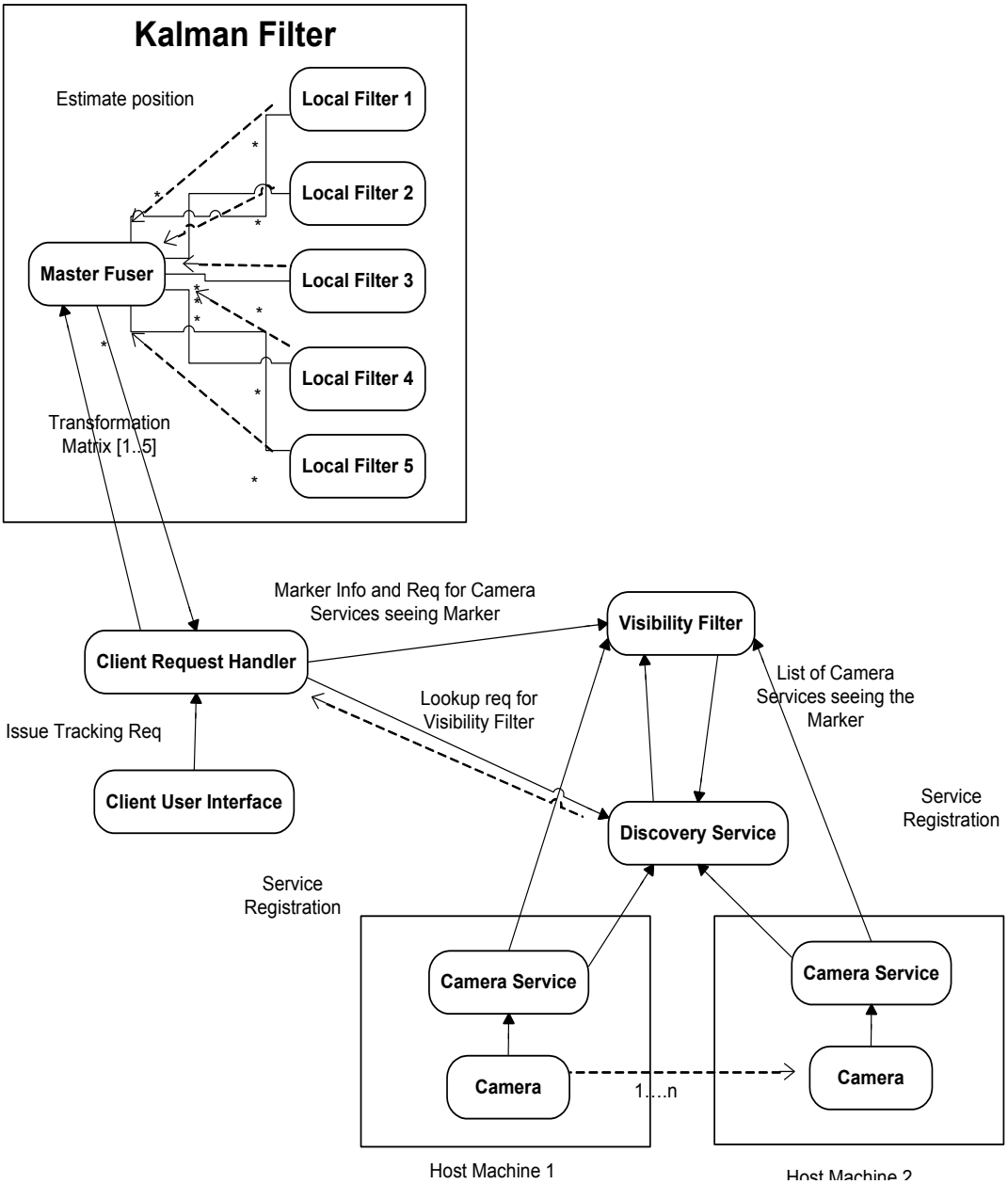

Figure 3.1 e-DTS Architecture [2]

The system consists of Cameras, Camera Services, Filter Proxies, the JINI Discovery Service, Tracking Markers, and a Client. The Cameras are interfaced via a USB connection and are accessed by utilizing the ARToolkit API [3] using the JAVA binding jARToolkit [31]. This information is then directly tied to a Camera Service in a one-to-one relationship. This Camera Service is then joined with a Filter Proxy in a many-to-one relationship. This Filter Proxy is responsible for making the data ready for consumption as well as providing a single interface in which the client can retrieve the Camera data. In the e-DTS, the Client Request Handler serves as the Tracking Client and receives tracking requests from the user. These requests can then utilize the power of a Kalman-based fusion technique in order to retrieve data from a Filter Proxy and then fuse the data retrieved in an effort to provide more accurate vision based tracking.

3.2 e-DTS Components

The following subsections will go into more detail regarding the various components of the system described above. A more detailed treatment of these components can be found in [1] and [2].

3.2.1 Camera and Camera Service

The cameras being utilizing in the e-DTS are inexpensive Web-cameras. Two different brands of cameras (Logitech QuickCam [32] and Micro Innovations Basic Webcam) are used in the setup of the e-DTS. These cameras form the basis for tracking an object and allow for a pattern attached to the object to be seen and recognized [3]. The Camera Service is a Web-based JINI service that interfaces with the physical cameras and provides access to the details provided by the ARToolkit API [3]. The Camera Service is responsible for locating and registering itself with the JINI Lookup Service in an effort to allow it to be discovered and then consumed by a client. The Camera Service provides a 4 x 4

transformation matrix that includes the estimated pose, i.e., the position and orientation, of the object being tracked as it relates to a known coordinate system. This information then can be used in the fusion process to allow for a more accurate tracking estimation of the object as it moves through the environment.

3.2.2 Discovery Service

The Discovery Service provides the means of locating other services within the e-DTS. More specifically, the process of discovery involves finding services that are available within the network and then establishing a connection with a located lookup service. The e-DTS discovery component has been implemented utilizing JINI architecture. JINI, is a network architecture for the construction of distributed systems in the form of modular co-operating services [4]. Within the JINI framework there are three types of discovery protocols that are available to be consumed, those being: Multicast Request Protocol, Multicast Announcement Protocol, and Unicast Discovery Protocol. This can be further simplified by combining the first two protocols listed under the name Multicast Discovery Protocol.

The Unicast Discovery Protocol utilizes an underlying TCP connection with the host running the JINI service. This discovery protocol is best suited for a well-known and established environment where each host knows the location of the JINI service and that a reliable connection is available. The e-DTS relies on the Unicast Discovery Protocol. Once a JINI Lookup Service has been established, a Camera Service can register itself with the Lookup Service, thus allowing it to be discovered using a known protocol. The known protocol is predetermined prior to an initial setup and it is expected to be reliable and is the driving factor in this tracking design.

The Multicast Discovery Protocol utilizes an underlying UDP connection which means that discovery is dynamic and is flexible to a changing environment.

In a Multicast environment, a service first will attempt to locate available JINI Lookup Services within a given LAN. After it has located available Lookup Services, the service will register its Proxy with the Lookup Service(s) so that any client can consume the service. For the Client it simply broadcasts a message seeking to discover any JINI Lookup Services on the local network. If any are found then it attempts to find any services which match its request. If a service is found then the Proxy that was provided by the Service is downloaded and consumed by the Client.

The e-DTS relies on the Unicast discovery as a means to communicate with the JINI lookup services. In the e-DTS, each Camera Service is provided with the network location of the JINI lookup service and it is then able to communicate with that particular lookup service. While this method does provide the benefits of a known lookup location, it has the downside that only the known and existing lookup service can be interacted with. This does not provide flexibility to the e-DTS, because if the lookup service is no longer active or responding then the e-DTS is unable to track an object.

3.2.3 Tracking Markers

Tracking Markers are images or patterns that can be easily recognized and tracked by the ARToolkit API. The marker itself is a simple and distinctive pattern which is recognizable by the ARToolkit API [3]. A tracking marker, or fiducial, usually consists of contrasting colors (i.e., black and white) that make it easy for computer software to distinguish and properly identify the pattern from other objects within the scene.

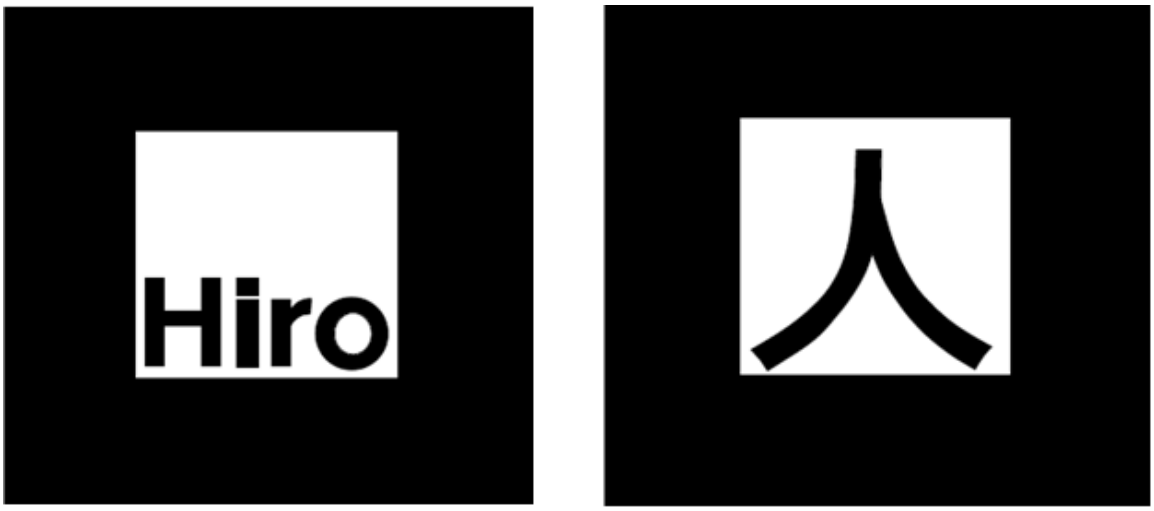

Figure 3.2 Fiducial Markers - pattHiro and pattKanji [3]

A tool provided with the ARToolkit API [3] allows for a pattern to be created and subsequently be used and tracked. Once an image pattern is provided to the tool, the patterns are converted to data files in an effort to be utilized by the API in order to correctly identify them with the visibility frame provided by the physical camera. In the e-DTS, the object being tracked is associated with such a marker.

3.2.4 Visibility Filters (VF)

The purpose of the Visibility Filter in the e-DTS is to serve as a proxy between a Client object and the JINI Discovery Service – using the Unicast Discovery Protocol. The VF replaces the concept of a Proxy Object in the traditional sense of a Multicast Discovery Protocol. When a Client object requests the access of a particular Tracking Service, it will communicate with the Visibility Filter which will in turn proceed to do a lookup to find all registered Camera Services. The VF will then return the information regarding any Services that are currently registered and actively seeing/tracking the marker pattern as it moves throughout the environment. Therefore, the Tracking Client need only

communicate with the VF as it is responsible for maintaining all of the data for any Camera Services associated with it. The VF also allows for work to be offloaded from the Camera Service to a VF created camera thread in an effort to reduce the overhead between communication and processing.

3.2.5 Kalman Filter

The Federated Kalman Filter is based upon a design by Carlson in [19] for a Federated Kalman Filter for use in a distributed environment. This setup consists of the sensors, in this Camera Services and their Cameras, Local Filters, and a Master Fuser responsible for predicting the position of the object. The Master Fuser uses fine-tuned square root algorithms for performing the data fusion and subsequent estimation of position. The Federated Kalman Filter process is shown below in Figure 3.3.

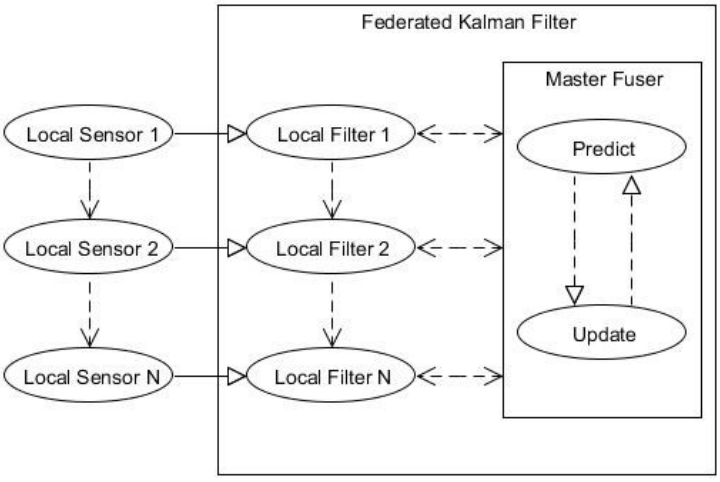

Figure 3.3 Federated Kalman Filter [19]

The Kalman Filter uses recursive estimation of the state of process, in a noisy environment from one or more sensors. The Kalman Filter requires that two models to be present: a dynamic model that allows for the prediction of the state at a given time based on the state at a different time, and a measurement model in which the state information is related to observations. In the e-DTS, the

transformation matrices of all ‘Seeing’ Camera Services are provided, via the Tracker Client, to the Master Fuser which begins the filtering activity in an effort to provide an accurate estimation of the position of the object within the environment.

3.2.6 Tracking Client

The Tracking Client, associated with a specific marker, is a front-end user application that interacts with the VF in an effort to provide tracking information regarding eligible fiducial markers within the environment. The Tracking Client receives information regarding the position of the marker with respect to an established coordinate system and then can in turn send the information to the Kalman Filter which will provide Kalman-based fusion on the data set. The Tracking Client also provides the means to collect statistics on the overall performance of the e-DTS along with data regarding the various services. This data is made available in an effort to keep the user informed upon the happenings of the e-DTS as a user may want to receive information regarding what Services are currently available for consumption. A sample interaction with the Tracking Client is as follows:

- A Tracking Client begins execution by locating the JINI Lookup Service provided; this is done by submitting a discovery request with the location name provided to the component.
- The Tracking Client registers any known and provided patterns with the service in an effort for the Cameras and their Camera Services registered with the Lookup Service to respond to tracking inquires.
- The Cameras begin tracking registered patterns found within the environment; if one is identified by the tracking API then a flag is set on the Camera Service to indicate that the service is currently

'Seeing' the registered pattern. This allows the VF to quickly locate and identify only those 'Seeing' services.

- The Tracking Client can then establish communication with any Visibility Filters found registered with the Lookup Service. If one is found the Tracking Client can retrieve relevant tracking data.
- Once the Tracking Client retrieves information regarding a registered pattern it can send this information to the Kalman Filter in an effort to provide Kalman-based fusion techniques to the data.
- The Kalman Filter will then return this data to the Tracking Client which will then make the data available to the user.

This process is repeated continuously while a camera within the tracking system is able to recognize and see a registered pattern.

3.3 Implementation

This section will discuss the methods for implementing the e-DTS as described above. ARToolkit [3] provides an API for the marker recognition and tracking – this API has been adopted and a Java-binding has been provided in the form of the jARToolkit [31]. This Java-bound API allows for each of the Camera Services to receive the optical information from the physical cameras themselves. The Camera Services can then provide the transformation matrix of the Marker with regards to the Global Coordinate System as provided by the Camera. For each physical Camera involved in the implementation, a corresponding Camera Service must be present. In addition to this at least one VF must also be present. The machines are connected via a 10 Mbps LAN connection. The Tracking Markers have been mounted to heavy card stock in an effort to provide a rigid and reliable pattern.

3.4 e-DTS 2.0

The e-DTS provides a generic and tested framework for a vision-based indoor distributed tracking system. The framework could be improved with dynamic discovery, the concept of a shared and global coordinate system, communication between various different worlds as well as the hand off between cameras and the tracking details, improved camera calibration, clock synchronization between the machines involved in the e-DTS, and finally, QoS of parameters for the Cameras and their services. By implementing dynamic discovery the system could handle faults more gracefully while providing the ability to dynamically contract and expand based upon the environment in which it is located in. Through the implementation of a shared and global coordinate system the various Camera Services could provide tracking as the object moves from one known coordinate system to another seamless with appropriate handoff and transformation between environments. Improved camera calibration in turn would allow for the Cameras to provide a much higher degree of accuracy when it comes to estimation the location of the object with respect to its environment. Improved clock synchronization would improve upon the accuracy of the fusion process by allowing a greater sample of data to be acted upon by the Kalman Filter. By utilizing QoS parameters as a means for Camera Service selection and consumption, the system will be able to provide more accurate results based upon Camera specific parameters related to the fusion and estimation process. The following sections will describe these enhancements in more detail and provide quantitative data of these enhancements being implemented in the form of the e-DTS 2.0.

3.4.1 Reasons for Enhancements

As indicated earlier, the e-DTS system is based upon the Unicast Protocol [2]. This does not allow for dynamic discovery of services within the local network. Also, in the e-DTS, the location of the JINI Lookup Service must be

explicitly stated to each Camera Service. In addition to this, the JINI Lookup Service must be actively running and accepting queries. In this scenario, if the JINI Lookup service is interrupted or terminated, a subsequent error will be generated by the e-DTS and it will terminate as no further communication can take place as it relies on a central location. Due to these shortcomings, the e-DTS is not flexible when it comes to change in the Lookup Service and is unable to handle faults with regards to discovery and communication.

In addition to this, in the e-DTS there is no Global Coordinate System shared between the various cameras involved in tracking. Therefore, each camera must be first calibrated with respect to another camera in an effort to determine its location. This does not allow for a seamless handoff between cameras as the object is moved within the environment of the e-DTS. Also, as a result of not sharing of the coordinate systems, there is no way to achieve coordinate handoff as an object transitions from one tracking environment to another.

A critical goal of any distributed system is the need and desire for time synchronization. Without such synchronization two processes or queries could become isolated due to timing constraints. Because tracking demands real-time processing, there is a need for the clocks of the various machines in the e-DTS to be synchronized in an effort to offset clock drift. Such a synchronization is necessary as readings from individual cameras are fused together to obtain the position of the tracked object. This synchronization is also critical when attempting to perform Kalman-based fusion on the data sets returned in the tracking process. Kalman-based fusion was introduced into the e-DTS, but without sufficient clock synchronization the clock drift negatively impacted the effectiveness of the Kalman fusion.

Finally, at the heart of any tracking application is the calibration of the trackers themselves, in this case the Cameras. The calibration tools provided with the ARToolkit API, used by the e-DTS, are not designed to be used for tracking applications [3]. The more accurate the calibration the more accurate the

tracking results will be. Therefore, a better calibration tool or method is needed in an effort to improve the overall results of the Camera tracking.

The e-DTS 2.0 addresses all these limitations of the e-DTS as indicated in the next section.

3.4.2 Proposed Enhancements to the e-DTS

The e-DTS utilizes the Unicast Discovery Protocol. This protocol is based upon the idea that each client knows the location of the JINI Discovery Service. While this is a good method for a known and stable environment, it does not suite a dynamic environment in which the e-DTS could be deployed in. In addition to this, the Unicast Protocol does not allow for dynamic discovery of additional JINI Discovery Services if one fails, thus, the current e-DTS is not fault tolerant with regards to service discovery, registration, and retrieval. The proposed enhancement includes modifying the e-DTS to utilize the Multicast Discovery Protocol as the means to discovery JINI Discovery Services within the local network. This allows for dynamic discovery and provides a fault tolerant system. Another additional enhancement in terms of network and service discovery is to allow for the user to specify the location of the JINI Discovery Service as a means to allow for external or remote tracking outside of the LAN in which it resides on.

The e-DTS does not handle clock synchronization and thus the clocks within the distributed system will gradually drift over time. This concept of clock drift is noted in [20]. The proposed enhancement to the e-DTS is to provide a method for each machine that hosts the Camera Services to be periodically synched with a global time value. This periodic synching of these clocks will help to reduce the amount of clock drift found when attempting to fuse the results provided by Camera Services currently seeing the Tracking Marker and hence the object. The proposed time synchronization client is a third party application,

Windows Time Agent [9], with the time servers of the United States Government and IUPUI being used as the master time.

The e-DTS utilizes the provided calibration tool from the ARToolkit API [3]. This tool will suffice for basic camera calibration, however, improvements in camera calibration can be made with the assistance of a tool provided by Cal Tech [11]. This tool in conjunction with MatLab allows for a more accurate camera calibration.

As a tracked object is moving through an environment, the need for sharing and communicating the coordinate system being used is vital in an effort to provide an accurate estimation of the current pose of the object. This will also allow for a more fault tolerant system as if one Camera Service should fail the remaining Camera Services can continue to communicate and share tracking information based upon the known coordinate system.

3.4.3 Enhanced Features of the e-DTS 2.0

This section discusses the improvements that the e-DTS 2.0 proposes to overcome the limitations of the e-DTS (described in the previous section). Only the entities that were modified to create the e-DTS 2.0 from the e-DTS are described here for the sake of brevity.

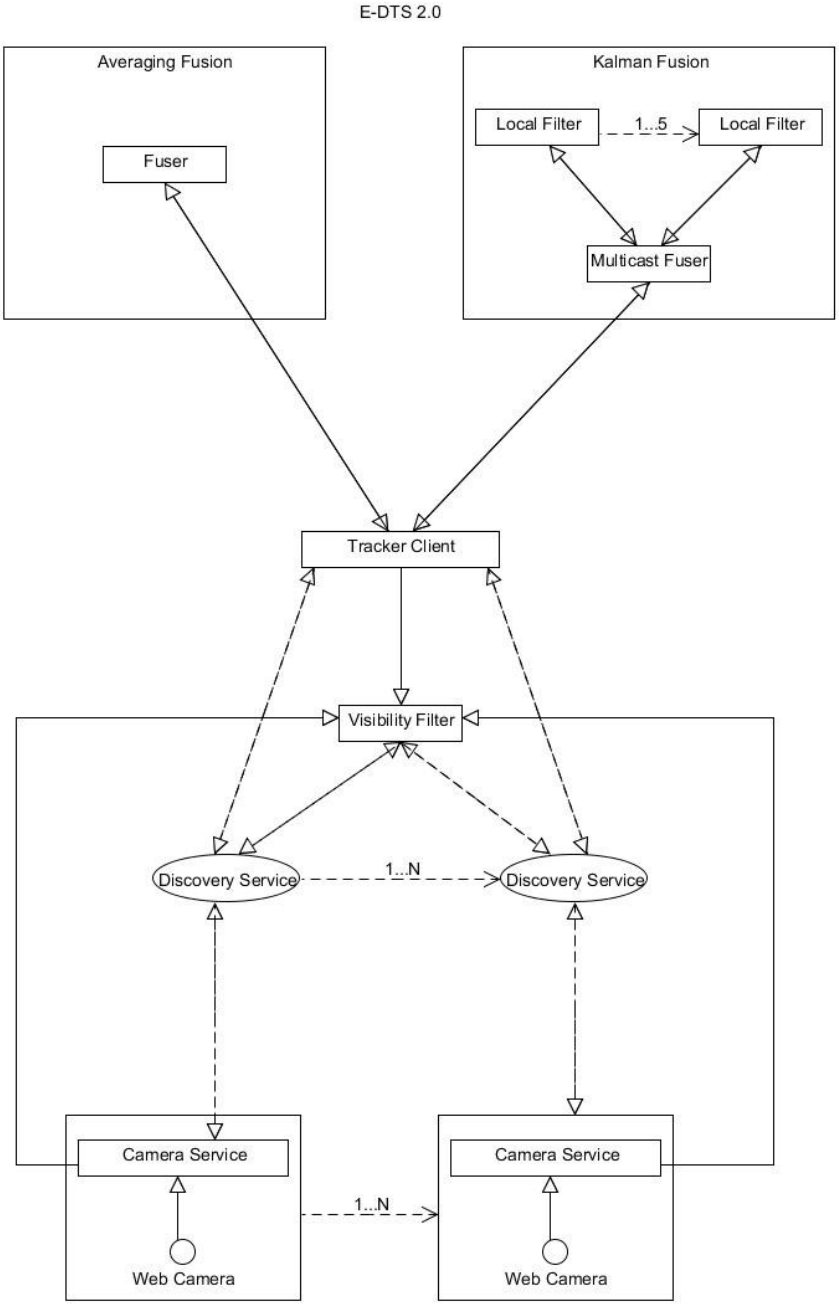

Figure 3.4 e-DTS 2.0 System Architecture

3.4.3.1 Camera Service

The Camera Service component of the e-DTS 2.0 has been modified to utilize the power of multithreading in an effort to constantly scan the local network for any JINI Lookup Services present. If a lookup service is found, then the Camera Service will register itself with it and store the location of the lookup service. This is done in an effort to provide fault tolerance to the e-DTS 2.0 by allowing it to re-register with any previously known lookup services if a fault should occur. The dynamic nature of the discovery and communication also allows for inter-camera communication and the sharing of coordinate systems. This will be discussed later within this section.

The second enhancement to the Camera Service involves a background thread that provides the ranking of that Camera Service. The Camera Service maintains a parameterized list of QoS attributes that decides the rank of each Camera Service. These attributes can be altered or adjusted by the user but begin with a base set of: *Display Resolution*, *Location to the Origin*, and *Clock Drift*. The Display Resolution is provided by the Physical Camera itself and is static in that the resolution of the camera does not change during the duration of the Camera Service. The Location to the Origin is based upon the Camera's physical location with regards to the Global Coordinate Systems origin. This allows for the selection of a Physical Camera and the associated Camera Service to be based upon its location. The final preset parameter is the drift between the clock associated with the Camera Service and the Windows Time Agent Service. This allows for the selection of the Camera Services to be based upon the associated Clock Drifts. These three parameters are selected on the basis that each provides a different aspect of the QoS for the e-DTS 2.0. The first parameter, Display Resolution, is tied directly to the Physical Camera itself. The second parameter, Location to the Origin, is tied directly to the shared Global Coordinate System implemented. The third parameter, Clock Drift, is tied to the concept of clock synchronization within a distributed system. The Camera Services are ranked by the Visibility Filter (VF) via a ranking thread; this ranking

is then communicated back to the Camera Service which stores its associated rank. This ranking is constantly being updated and propagated to the participating Camera Services in the background by the VF. This allows for the rankings to remain fresh with respect to the performance of the overall system. This allows for broad coverage of QoS for the e-DTS 2.0.

3.4.3.2 Visibility Filter (VF)

The VF has been modified to utilize the Multicast Protocol. This is achieved by using multithreading to provide continuous background scanning, using the Discovery Listener from the JINI API, of the local network to locate any available JINI Lookup Services. If a Lookup Service is found, then the VF will register itself with it and check to see if any Camera Services are currently registered with that particular Lookup Service. If there are Camera Services registered with the Lookup Service, then the VF will collect the information on these services. This allows the VF to serve as a proxy of the Tracking Client.

3.4.3.3 Tracking Client

The Tracking Client is subdivided into two separate modules within the same client in the e-DTS 2.0. This separation allows two different techniques for data fusion: the simple averaging fusion and the Kalman-based fusion technique. In addition to this, the Tracking Client employs the Multicast Protocol to dynamically discover any Camera Services that are located within the local network. The Tracking Client has also been modified to allow for the user to provide a known location of a JINI Lookup Service located outside of the LAN in an effort to provide remote tracking. The Tracking Client also takes into account the ranking system established for the Camera Services based upon the QoS parameters. The Tracking Client uses these QoS parameters when selecting services for the data fusion. The following is a breakdown of the QoS parameters

provided and how they are utilized by the Tracking Client in determining Camera Service selection.

The first QoS parameter of discussion is that of Camera Resolution. This parameter is populated by the underlying jARToolkit API when the physical camera is initialized by the Camera Service. The default value for the resolution of the physical cameras is 320 x 240. This is determined to be a standard for all of the cameras within the e-DTS 2.0. This parameter value, however is not static for each camera and can be altered or adjusted – however, once an associated Camera Service is registered with the JINI Lookup Service, this parameter cannot be changed.

The second QoS parameter of discussion is that of Camera Distortion. In the e-DTS 2.0, the physical cameras utilized have a wide field of vision [2]. This allows for high distortion with the image being seen and tracked. This parameter allows for the Tracking Service to distinguish and select only those cameras with low distortion when attempting to perform fusion.

The third QoS parameter of discussion is that of Camera Frame Rate. This parameter allows for the physical camera itself to provide the frame rate at which it is able to capture an image within its view. This parameter is established when the physical camera is initialized and cannot be changed once the associated Camera Service is registered with the JINI Lookup Service. This parameter, however, is not static and can be changed.

The fourth QoS parameter is the Relative Location of a Camera. This parameter is related to the location of the physical camera with respect to the object being tracked. The closer a physical camera is located with the tracking object the higher probability is that the Tracking Client will use its data. This parameter is dynamic and is constantly changing based upon whether a Camera is seeing/tracking the object.

The fifth QoS parameter is the Clock Drift. The Clock Drift is related to the varying difference between the Tracker Client's system clock and that of the Camera seeing the object. The Tracker Client will select those Camera's that

have low clock drift, meaning that the system clock value as provided by the Camera is identically or only slightly different than that of the Tracker Service.

The Tracker Client also provides the user the option to select which type of fusion they wish to perform in an effort to provide an accurate tracking location estimate of the objects in question. Once the user has selected which option they desire, simple averaging or Kalman-based fusion, the Tracker Client will then begin fusing results returned to it via the Camera Services. A sample interaction between the User, the Tracker Client, the Fuser, and the Camera Service is as follows and is displayed in Figure 3.5:

- The User selects either simple averaging technique fusion or Kalman-based fusion. In the background the Tracking Client is searching for local JINI Lookup Services in an effort to locate any Camera Services present.
- Once a Camera Service is found, the Tracking Client attempts to retrieve the data filtering based upon the QoS parameters that have been established.
- Once the Tracking Client has filtered the data received, it will then attempt to perform any fusion necessary. Fusion takes place when two or more cameras are currently seeing the same Tracking Marker.
- Once the Fusion has been performed the results displaying the current estimated position are returned to the user of the relative physical location of the object with respect to the Global Coordinate System.

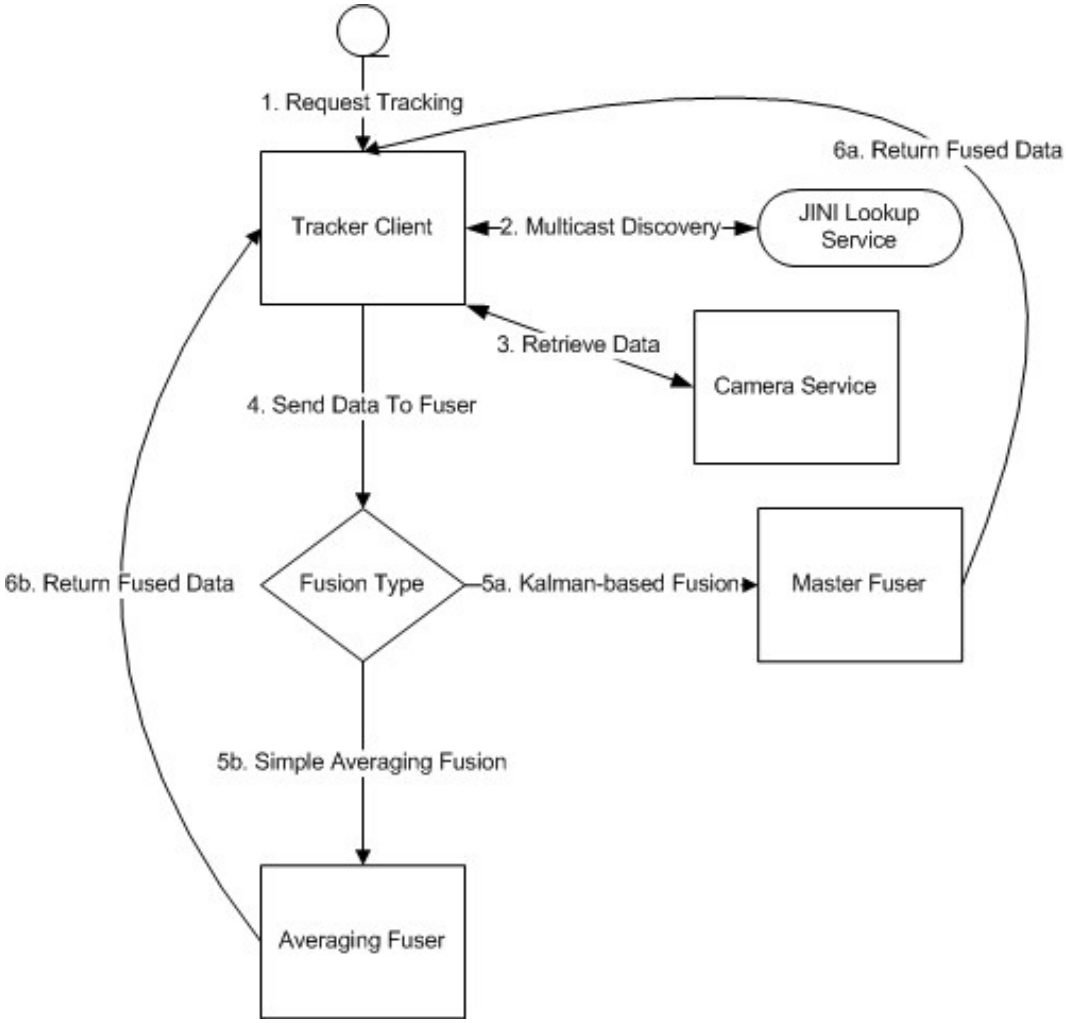

Figure 3.5 Tracking Process in the e-DTS 2.0

In the case of multiple cameras ‘Seeing’ an Object, the Tracker Client can provide the coordinate transformation and handoff as the Object moves from one known environment to another. This is done by the implementation of Spatial Relation Graphs (SRG) that are maintained by both the Camera Services and the Tracker Client. When the Tracker Client determines that a coordinate transformation and handoff must be done it selects a Camera Service from each environment, based upon QoS parameters and rank, and performs the coordinate transformation. Once this transformation is complete, the Tracker Client sends a message to the two participating Cameras Services in order for the Camera Services to be updated with the knowledge of the coordinate

transformation. This process, of updating, involves each of the two Camera Services to maintain a list of other known environments and their transformations with respect to the particular environment coordinate system. This is done in an effort to expedite the transformation process as future inquiries into coordinate transformation will not need the additional overhead as the Camera Services themselves will know of the coordinate transformation. Because the Tracker Client is aware of all 'Seeing' services, it is able to determine when coordinate handoff or a transformation should be done all the while using the information that the Camera Services themselves have learned by participation within the setup.

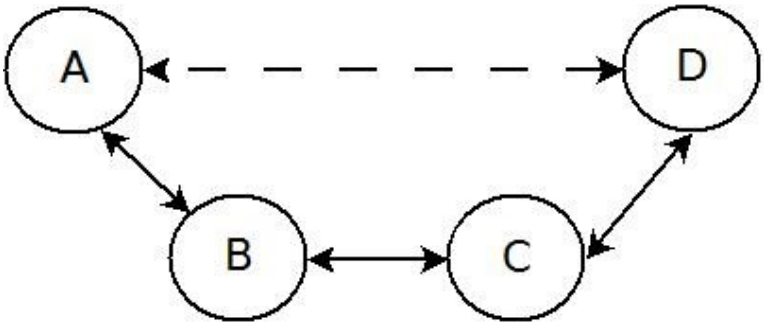

Figure 3.6 Graph for determining world transformation in e-DTS 2.0

In Figure 3.6 the nodes B and D represent the Cameras and their corresponding services; the node A represents an origin; and the node C represents a Tracking Marker. In the e-DTS 2.0, each camera is calibrated with respect to its own global coordinate system. Under the assumption that cameras B and D have been calibrated in different coordinate systems and are located in different worlds then there is no direct correlation between D and the origin A. Because of this, each camera will provide the pose of the object with relation to two separate coordinate systems rendering the fusion process useless in the case of two different worlds 'Seeing' the tracking object. However, since there is a known path from A → D via the nodes B and C then a concatenation of the three transformations A → B, B → C, C → D will yield a transformation for the direct path A → D. Therefore, with this knowledge the pose estimation from node

D can be converted with respect to the coordinate system established by the node A.

This section has provided the basis for the enhancements and features for the creation of the e-DTS 2.0. It has provided the details of how the proposed enhancements have been implemented into the existing e-DTS. The next section will discuss experiments run against the newly improved e-DTS 2.0 and the results that were gathered through these simulations.

CHAPTER 4 EXPERIMENTATION AND VALIDATION

This section provides a comprehensive discussion regarding the experimentation done with the prototype e-DTS 2.0.

The experimental setup involves ten Dell OptiPlex machines running Windows XP SP 3 with Pentium 4 CPU's connected on a 10 Mbps local area network connection. The Eclipse IDE was utilized as the Java development and debugging tool for the prototype of the e-DTS 2.0. Each of the ten machines contained two Web cameras, Logitech QuickCam and Micro Innovations Basic Webcam, attached via a USB port. This allows for twenty cameras to be setup and configured within the environment. The experimental environment also consists of a well-lit lab with the dimensions of 29.7 feet by 19.5 feet. Each of the twenty cameras has been calibrated prior to running the prototype of the e-DTS 2.0 and a customized calibration data file has been provided to each Camera Service to use.

4.1 Experiments to Test Dynamic Discovery

In a distributed system the components must be able to discover one another using one of the two following protocols: Unicast or Multicast. In the Unicast Protocol the component must explicitly be told of the location of the service it wishes to communicate with while in the Multicast Protocol the component itself can send out a message requesting that any services present to respond back with their given information. As indicated earlier the current prototype of the e-DTS is built upon the Unicast Protocol where each Camera Service is aware of the JINI Lookup Service's location. This presents two

problems: first the Camera Service must be explicitly told the location of the JINI service, and second if either service should terminate or become unavailable the inability for re-establishing a connection is not possible. With the Multicast Protocol the services can self-learn of the environment in which they are currently existing in which provides the benefit of self-healing when a problem with discovery occurs.

4.1.1 Experiments to test the functionality of the Multicast Protocol

When discussing the implementation of the Multicast Protocol in place of the existing Unicast Protocol no discussion is complete without providing previously established metrics of measurement. Many metrics established by [1] can be reused or altered to conform to the needs of the e-DTS 2.0. In addition, a new metric, End-To-End Discovery Time (EEDT), has been created for the e-DTS 2.0. This is a variation of the End-To-End Response Time (EERT) that focuses solely on the time required for the JINI Lookup Service to be discovered and subsequent registration to take place. This metric was introduced as a means to quantify the total time it takes for the discovery process to be completed. In this particular experiment, the EEDT is defined further as the total time taken for a service to discover any existing JINI Lookup Services, register itself with any services discovered, and finally, to lookup any other services currently registered.

This first experiment is to test the EEDT for the e-DTS 2.0 when using the Multicast Protocol in comparison with the Unicast Protocol found and used in [2]. The test involved a stationary marker within view of ten of the cameras within the environment. For this first experiment, it was determined to use a static marker to identify and perform the maximum number of fused cameras thus creating a steady stream of communication and discovery by the Tracker Client and the Camera Services.

As discussed by [2] the e-DTS utilizes the Unicast Discovery Protocol in an effort to locate and register Camera Services. The e-DTS 2.0 built upon this idea of discovery and subsequent registration by adding parameters to the service registration. This allows, as will be discussed later in the Environmental Worlds section, the concept of rank and order amongst the Camera Services. This provides for a more streamlined and efficient method of organizing a hierarchy within the Camera Service world and allows for sharing of coordinate systems.

Therefore, an addition of a simple and light-weight approach utilizing the Multicast Protocol needs to be implemented. This was achieved by using only the key elements of the DTS: tracking, pattern recognition, fusion, and statistical and tracking information. This allowed the e-DTS 2.0 to achieve the desired results by implementing both a Camera Service, a Camera Proxy object – used for the client side communication, and a Tracking Client. The Camera Service made available via the Camera Interface and its proxy object the ability to communication the transformation matrix found by the jARToolkit API. The Tracking Client was then able to use this information to fuse any results found and display the tracking information accordingly.

Table 4.1 Average EEDT

Protocol	**30 seconds**	**60 seconds**
Unicast	2.1173	3.1982
Multicast	2.1611	3.2226

In Table 4.1 the results of the experimentation are shown with the Unicast Protocol only slightly outperforming the Multicast Protocol in terms of EEDT. This was expected as in Unicast Discovery the location of the JINI Lookup Service is

known while in the Multicast Protocol the network must be scanned in an effort to find an eligible JINI Lookup Service. This demonstrates that the Multicast Protocol can be used in place of the Unicast Protocol in the e-DTS 2.0 with only a slightly higher overhead with respect to the average discovery time.

A second experiment to evaluate the Average EERT between the Unicast Protocol and Multicast Protocol, shown in Table 4.2, demonstrated that the average additional overhead created by the Multicast Protocol was 1.5 milliseconds.

Table 4.2 Average EERT

Protocol	**Average EERT (milliseconds)**
Unicast	36.1
Multicast	47.6

Once again, this was expected as the EEDT is a component of the EERT and thus the Multicast Protocol would introduce additional overhead. These experiments provide a quantitative approach to demonstrating that the Multicast Protocol can be beneficial while yielding only slightly higher average overheads with compared to the Unicast Protocol used in the prototype of the e-DTS described in [2].

4.2 Camera Calibration

In an effort to improve tracking extensive evaluations using various calibration tools was performed in an effort to provide the most accurate position estimation of an object. Using tools, provided in [11] using the Matlab Software Suite, all of the cameras within the e-DTS 2.0 were calibrated with respect to their environment. A simplified calibration tool is provided with the ARToolkit

software package. Because inexpensive web cameras were being used, the overall quality and resolution is limited by the devices themselves.

An experiment was created to test the overall accuracy, in terms of estimated error between the estimated and actual physical location of the calibration pattern, of the tool, provided in [11], versus that provided with the ARToolkit software package.

Table 4.3 Average Calibration Error

ARToolkit Calibration	**Alternate Calibration Technique [11]**
35.01270 mm	14.6573 mm

In Table 4.3 it can be seen that the average calibration error calculated using the technique described in [11] is considerably less than that generated through the use of the tool found in the ARToolkit software package. By using this alternate technique, the ARToolkitPlus API must be incorporated into the e-DTS 2.0 as a means to utilize the data provided through the use of [11]. This demonstrates the importance of the tool and technique selected during camera calibration and shows that an improved technique used in the e-DTS 2.0 can greatly improve camera calibration with respect to that found in the ARToolkit Calibration Tool used in the e-DTS described in [2].

4.3 Environmental Worlds

In a distributed tracking system the core concept is its inherent dynamic nature. Therefore, the tracking system must be designed in such a fashion that both services and objects can come into and leave environments all the while receiving accurate tracking information while they are seen within the environment.

For this particular experiment the e-DTS 2.0's design and an initial setup has been based on the common hierarchy similar to that found in the military. In this hierarchical setup the e-DTS 2.0 uses the concept of groups, first level services, and second level services. The e-DTS 2.0 experimental world has been divided into six groups, each of these groups containing one first level service and the remainder being second level services – each of the second level services ranks itself according to its "status" within the group. These groups then have the ability to talk to one another and share coordinate systems throughout the tracking of an object within their world.

As mentioned previously, the e-DTS 2.0 takes a hierarchical approach to encompass the concept of multiple environmental worlds. For this experiment, the system was divided into six different groups within one environment. Five of the six groups contained a total of four camera services and visibility filters with the sixth group consisting of a mobile camera on a separate machine with its own visibility filter. For this initial experiment, the roles were pre-determined and assigned within each group.

Table 4.4 Service Check Frequency (Dead Service)

Service Check Frequency	**< 1 second**	**1 second**	**15 seconds**	**30 seconds**
Total Time (milliseconds)	21084.4	192.990	109.7642	71.0930

In this experiment it was determined that there were two constraining parameters for the overall tracking of an object, those being time and accuracy. For each of these constraints a series of experiments were ran to test and categorize these parameters in an effort to provide a group for potential applications.

4.4 Clock Synchronization

For the experimentation with clock synchronization, three available software tools were utilized when trying to tackle the issue of clock drift and clock synchronization in a distributed system. The three tools that were selected to use were: Atomic Clock, Domain Time II, and Windows Time Service. It was decided to use these three after an extensive search of available tools to aid the issue of clock drift among machines.

For each software tool selected a sample clock drift and synchronization client and server were executed for a duration of 48 hours with samples taken every 10 minutes and written to a text file. This duration was decided in an effort to provide an accurate and quality sized sampling of clock drift between the machines. After running the experiment the data was able to be collected and further analyzed by providing the average clock drift between synchronization attempts with the time server. In each instance one of the machines was established as the host and the “official” global time, if this host was unavailable then the time would be collected from the time would be attempted to be synched to the following two time agents, in the listed order:

1. time.nist.gov
2. ntp.iupui.edu

These experiments demonstrated that the Domain Time II Client worked the best in keeping the machines synched to the global time. Once all of the data was collected an averaging could be performed to demonstrate the average of the clock drifts of the sampled machines. This allowed for a better understanding of the data that had been collected. As shown in the results in Table 4.5:

Table 4.5 Clock Drift Averages

Tool	Average Clock Drift (milliseconds)
Atomic Clock	665.7693
Domain Time II	7.56994
Windows Time Service	2035.04

In Table 4.5 the average clock drift was calculated by taking an average from the experiment with each tool and then comparing each tool to one another. As shown, the Domain Time II client performed the best with regards to maintaining clock synchronization. By utilizing these tools the e-DTS 2.0 can provide a greater degree of clock synchronization than that of the e-DTS proposed in [2], due to its absence of a clock synchronization method.

Upon further analysis of the data shown in Table 4.5, it was decided to gather more information on the Domain Time II tool by analyzing the maximum and minimum clock drift errors between samples. This was done to provide QoS metrics, i.e. the minimum and maximum clock drift found when using the tool, regarding this tool when implemented within the e-DTS 2.0 This further analysis and experimentation is shown in Table 4.6.

Table 4.6 Domain Time II Clock Drift

Machine:	1	2	3	4	5	6	7
Max Error	19	29	24	20	48	32	20
Min Error	0	0	0	0	0	0	0

This reason behind this experiment was due to the fact that the Domain Time II client proved to provide the lowest average clock drift between readings, thus making it the best choice for clock synchronization in the e-DTS 2.0. It was found that between the various machines involved that the maximum clock drift error was 48 milliseconds and the minimum was 0 or the absence of clock drift between the client machine and the host time server. With this knowledge the Domain Time II Client was applied to all of the machines within the e-DTS 2.0 in an effort to provide a more reliable and universal time.

4.5 Fusion and Tracking

The goal of the following experiments was to demonstrate the accuracy of the e-DTS 2.0 when using fusion techniques to estimate the actual position of the Tracking Marker.

The first experiment to test the e-DTS 2.0 implemented a simple average based fusion, first described in [1]. This averaging technique combines all of the readings for seeing cameras and takes the average for the (x, y, z) coordinates and displays this result as the estimated position of the Tracking Marker. The marker was moved slowly in front of all 20 cameras with the Tracking Client performing the fusion. Upon receiving information of a Camera Service seeing the Tracking Marker, the Tracking Client will both display the position data provided by the Camera Service, the Tracking Marker ID and name, as well as the timestamp associated with that particular reading. This information is written to a text file to aid further or future analysis. Table 4.7 indicates the average error, in millimeters, in the tracking estimate, using simple averaging fusion, for each of the three axes for the e-DTS 2.0.

Table 4.7 Estimated Error in Averaging Fusion (Millimeters)

Axis	X-axis	Y-axis	Z-axis
Average Error (millimeters)	59.9810	316.4367	114.8319

The actual physical position of the tracking marker was recorded and then using the results provided by the Tracking Client the error in estimation was calculated. Finally, these errors for the three axes were summed and an average was taken. This experiment demonstrates the average error found when attempting to estimate the position on an object with respect to its physical environment. As noted in [2], if the readings are subject to a constant systematic error then this knowledge can be taken to the tracking environment and used accordingly.

The second experiment to test the prototype of the e-DTS 2.0 implemented a Kalman-based fusion technique, first described in [1]. The marker was once again moved slowly in front of all 20 cameras with the Tracking Client performing the Kalman-based fusion technique. Table 4.8 indicates the average estimated error, using the Kalman based fusion technique, when tracking a marker as it moves through the environment.

Table 4.8 Estimated Error in Averaging Fusion (Millimeters)

Axis	X-axis	Y-axis	Z-axis
Average Error (millimeters)	16.1141	150.3517	99.9184

As can be seen, from Table 4.8, the calculated average error for the Kalman-based fusion technique greatly improves over the calculated error found in the simple average technique. Because of the clock synchronization in the e-DTS 2.0, a larger quantity of tracking data is available for performing the fusion. These experiments demonstrate that by using Kalman-based fusion the overall tracking accuracy can be improved over that of using simple averaging fusion techniques. This ability to track an object and give an estimated position with respect to its location within the environment is consistent with what was found in [2]. However, with the addition of a clock synchronization method and improved camera calibration, the e-DTS 2.0 provides the ability for a great quantity of data to be fused by the Tracker Client and subsequent Fusers, thus, a more accurate estimation of an object with respect to its location within the environment can be provided.

4.6 Coordinate System Handoff and Transformation

For the experiment for Coordinate System Handoff and Transformation the group of twenty cameras were split into two subgroups with ten being classified in World #1 and the other ten being classified in World #2. The Tracker Client was started and a Tracking Marker was introduced and slowly moved into World #1. For this particular experiment, the Tracking Marker was only moved upon the X-axis to minimize the overall calculations needed to verify the accuracy of the handoff in this preliminary experiment. As the Tracking Marker moved into ‘Seeing’ distance of World #2, the Tracker Client would perform the conversion and transformation from the data World #2 was supplying it with respect to the known world coordinate system in World #1. This data was then recorded and is shown in the Table 4.9.

Table 4.9 Handoff Transformation Error – Experiment #1 (Millimeters)

Axis	**X-axis**	**Y-axis**	**Z-axis**
Error (millimeters)	5.0231	120.1515	280.1742

Once the Tracking Marker moved out of the 'Seeing' distance of World #1, the handoff to World #2 was completed by the coordinate system for World #2 being used. The experiment was then tested again, this time moving from World #2 into World #1 which is shown in Table 4.10.

Table 4.10 Handoff Transformation Error – Experiment #2 (Millimeters)

Axis	**X-axis**	**Y-axis**	**Z-axis**
Error (millimeters)	630.1823	620.2250	270.0636

This tracking experiment can further be shown in Figure 4.1 shown below, which outlines the concept and the effects of moving the Tracker Marker from one coordinate system to another.

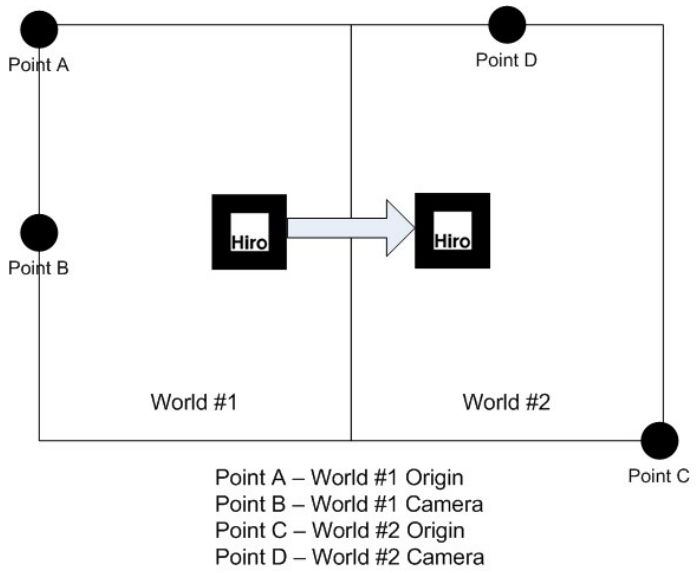

Figure 4.1 Transformation and Handoff Experiment

In these two experiments it can be shown that an additional error is created during the transformation process. However, this process does allow for coordinate transformation and handoff, something not found in [2] and because of its implementation in the e-DTS 2.0 demonstrates the ability to track an object as it moves from one coordinate system to another. As a result, the experiments, described in this chapter, demonstrate the ability of the e-DTS 2.0 to track an object as it moves through and amongst various environments.

CHAPTER 5 CONCLUSION AND FUTURE WORK

5.1 Future Extensions

Future extensions of this work could include but is not limited to:

- An exhaustive scalability study of the e-DTS 2.0.
- An exhaustive security and performance study of the usage of Multicast discovery within the e-DTS 2.0.
- Improved tracking and accuracy by utilizing other underlying API's.
- Implement the systems utilizing other sensors and trackers and explore the interaction between two different sensors.

5.2 Conclusion

In conclusion, this thesis has shown that the Domain Time II tool provides a much higher level of clock synchronization and accuracy when used in conjunction with the e-DTS 2.0 system and therefore, can greatly aid fusion and real-time processing. This thesis has also shown through experiments that Kalman-based Motion Fusion provides much higher accuracy when providing position estimates with respect to the Global Coordinate System using coordinate system handoff and sharing than previously accomplished in [2]. The simple averaging technique provides a much faster response time in terms of providing the pattern makers position, but suffers from a less accurate estimate. The e-DTS 2.0 is setup in such a manner that dynamic discovery of multiple worlds and

sharing the coordinate systems between the worlds is possible. A ranking system of the various cameras within the environment allow for this to be done efficiently and effectively. QoS parameters allow for selection of Cameras and Camera Services to take place by the Tracking Client thus allow for more accurate results to be generated through the fusion process as the higher ranked cameras are able to be selected and subsequently fused. This thesis has shown the importance and value in dealing with clock drift in distributed tracking systems and its impact on the fusion of data. It has also demonstrated the importance of QoS parameters in service selection and consumption as well as the importance of fine tuning of calibration methods in order to achieve a higher degree of accuracy. Finally, this thesis has shown through experimentation that the e-DTS 2.0 provides the capabilities to provide real-time vision based tracking in a dynamic environment.

LIST OF REFERENCES

LIST OF REFERENCES

[1] Girish G., Rajeev R., Mihran T., “Designing and Experimenting with a Distributed Tracking System.” Proceedings of the 2008 14th IEEE International Conference on Parallel and Distributed Systems, Pg. 64 - 71, 2008.

[2] Neha T., “e-DTS Enhanced Distributed Tracking System,” Purdue University, M.S. Thesis, 2009.

[3] Kato H., Billinghurst M., “Marker Tracking and HMD Calibration for a video-based Augmented Reality Conferencing System,” In Proceedings of the 2nd International Workshop on Augmented Reality (IWAR 99), 1999.

[4] Oracle, “Discovery and Lookup Services,” URL: http://download.oracle.com/javase/1.5.0/docs/guide/jmx/overview/lookup.html, Accessed 2010.

[5] Edwards, K., Rodden, T., “Jini Example By Example,” Prentice Hall PTR 1st Edition, 2001.

[6] Juhasz, Z., Sipos, G., “JGrid: A Jini Technology-Based Service Grid,” In European Community in Information Technology (ERCIM) News Number 59, Pg. 29 - 30, 2004.

[7] Bogucki, R., “Augmented Reality for First Response Scenarios: Navigational and Reference Device Based on Computervision Techniques,” University of New Hampshire, 2006.

[8] Chaos Software Group, “Atomic Clock Sync v3.0,” URL: http://www.worldtimeserver.com/atomic-clock/, Accessed 2010.

[9] Greyware Automation Products, “Domain Time II Windows Time Agent Version 4.1,”
URL: http://www.greyware.com/software/domaintime/instructions/misc/agent/, Accessed 2010.

[10] Microsoft Corporation, “Windows Time Service Technical Reference,” URL: http://technet.microsoft.com/en-us/library/cc773061%28WS.10%29.aspx, Accessed 2010.

[11] Bouguet, J., “Camera Calibration Toolbox for Matlab,” URL: http://www.vision.caltech.edu/bouguetj/calib_doc, Accessed 2010.

[12] Dorner, R., “Accuracy in optical tracking with fiducial markers: an accuracy function for ARToolKit, Mixed and Augmented Reality,” In ISMAR 2004 - Third IEEE and ACM International Symposium, 2004.

[13] Malbezin, P., Piekarski, W., Thomas, B., “Measuring ARToolKit Accuracy in Long Distance Tracking Experiments,” In ART02, 1st International Augmented Reality Toolkit Workshop, 2002.

[14] Claus, D., Fitzgibbon, A., “Reliable Fiducial Detection in Natural Scenes,” In Lecture Notes in Computer Science, Volume 3024, 2004.

[15] Claus, D., Fitzgibbon, A., “Reliable Automatic Calibration of a Marker-Based Position Tracking System,” In Seventh IEEE Workshops on Application of Computer Vision (WACV/MOTION'05) - Volume 1, 2005.

[16] Nagpal, R., Shrobe, H., Bachrach, J., “Organizing a Global Coordinate System from Local Information on an Ad Hoc Sensor Network,” In Proceedings of the 2nd international Conference on information Processing in Sensor Networks, April 22 - 23, 2003.

[17] Baroody, R., Al-Holou, N., Mohammad, U., Lahdhiri, T., Dayoub, I., "Scalability in a Dynamic Discovery Service-based Jini for the Next Generation vehicle network," In IEEE symposium on Computers and Communications (ISCC 2009), 2009.

[18] Chen, H., Joshi, A., Finin, T., “Dynamic Service Discovery for Mobile Computing: Intelligent Agents Meet Jini in the Aether,” In Cluster Computing Volume 4, Number 4, Pg. 343 - 354, 2001.

[19] Carlson, N.A., “Federated square root filter for decentralized parallel processors,” Aerospace and Electronic Systems, IEEE Transactions on Volume 26, Issue 3, Pg. 517 - 525, 1990.

[20] Lamport, L., Melliar-Smith, P., “Synchronizing clocks in the presence of faults,” In Journal of the ACM (JACM) Volume 32, Issue 1, Pg. 52 - 78, January 1985.

[21] Echtler, F., Huber, M., Pustka, D., Keitler, P., Klinker, G., “Splitting the Scene Graph Using Spatial Relationship Graphs Instead of Scene Graphs in Augmented Reality,” In Proceedings of the 3rd International Conference on Computer Graphics Theory and Applications (GRAPP), January 2008.

[22] Wagner, M., “Distributed Tracking with Multiple Sensors for Augmented Reality,” In Workshop “Virtuelle und Erweiterte Realität,” TU Chemnitz, September 27 - 28, 2004.

[23] Wagner, M., “Ubiquitous Tracking for Augmented Reality,” In Proceedings of the 3rd IEEE/ACM international Symposium on Mixed and Augmented Reality, November 2 - 5, 2004.

[24] Hashmi, N., Myung, D., Gaynor, M., Moulton, S., “A Sensor-based, Web Service-enabled, Emergency Medical Response System,” In Proceedings of the 2005 Workshop on End-To-End, Sense-and-Respond Systems, Applications and Services, June 5, 2005.

[25] Banaei-Kashani, F., Chen, C., Shahabi, C., “WSPDS: Web Services Peer-to-peer Discovery Service,” In Proceedings of International Conference on Internet Computing, 2004.

[26] Sivashanmugam, K., Verna, K., Sheth, A., “Discovery of Web Services in a Federated Registry Environment,” In 2004 IEEE International Conference on Web Services (ICWS 2004), July 6 - 9, 2004.

[27] Jimenez, J., “Design of a Distributed Tracking System for Camera Networks,” In TRUST Summer Undergraduate Program in Engineering Research at Berkeley (SUPERB), 2006.

[28] Pustka, D., Huber, M., Bauer, M., Klinker, G., “Spatial Relationship Patterns: Elements of Reusable Tracking and Calibration Systems,” In Proceedings of the 5th IEEE and ACM International Symposium on Mixed and Augmented Reality, October 22 - 25, 2006.

[29] Pentenrieder, K., Meier, P., Klinker, G., Gmbh, M., “Analysis of Tracking Accuracy for Single-Camera Square-Marker-Based Tracking,” In Third Workshop on Virtual and Augmented Reality of the GI-Fachgruppe VR/AR, September, 2006.

[30] Newman, J., Bornik, A., Huber, M., “Tracking for Distributed Mixed Reality Environments,” In The Engineering of Mixed Reality Systems, Human-Computer Interaction Series 2010, Springer London, ISBN: 978-1-84882-733-2, Pg. 251 - 273, 2010.

[31] Geiger, C., Reimann, C., Sticklein, J., Paelke, V., “JARToolKit - A Java binding for ARToolKit,” In Augmented Reality Toolkit, The First IEEE International Workshop, January 2002.

[32] Logitech, “Logitech Quick Cam,” URL http://www.logitech.com/, Accessed 2010.

[33] Scholl, B., Pylyshyn, Z., “Tracking multiple items through occlusion: clues to visual objecthood,” In Cognitive Psychology, 38, Pg. 259 - 290, 1999.

[34] Soto, C., Song, B., Roy-Chowdhury, R., “Distributed Multi-Target Tracking In A Self-Configuring Camera Network,” In Proceedings IEEE Conference Computer Vision and Pattern Recognition, Pg. 1486 - 1493, 2009.

APPENDICES

Appendix A. User Manual

The steps listed below are provided as a guide to setup the Enhanced Distributed Tracking System (e-DTS) 2.0. The e-DTS 2.0 has been tested to run on 32-bit processors with Java 5 or higher installed, running Windows XP/Vista/7. Due to the limitations of the ARToolkit API the e-DTS 2.0 will not run on 64-bit processors.

System Requirements:

- Windows XP SP2 or higher
- 32-bit Processor
- Java 5 or higher
- USB Web-Camera
- Local Area Network (LAN) Connection

1. Download the e-DTS.zip file that includes: the source code, batch files, JAR files, and sample tracking patterns from:

   http://www.cs.iupui.edu/~rrybarcz/eDTS

2. Unzip e-DTS.zip to any drive you desire. This will place a folder/create a directory, named DistributedTrackingSystem, containing all of the files needed to run the e-DTS 2.0.
3. Within the “utils” directory in the DistributedTrackingSystem folder you can find an executable file named DomainTimeII, this file will install the Domain Time II agent on your machine for clock synchronization. Once the application has been installed you can find the location on the Control Panel labeled Windows Time Service. At this point the Client can be configured to use the following two time server protocols: time.nist.gov and ntp.iupui.edu

4. The system can either be run via the command line or run within an IDE. The e-DTS 2.0 has been tested to run within the Eclipse IDE. If run from the Eclipse IDE skip to step 8, otherwise read below.
5. In order to run the e-DTS 2.0 from the command line you must first set the environmental variables. This can be done by going to the Control Panel, then to System, click on the Advanced Tab and click Environmental Variables. The following items should be present here in addition to any pre-existing values:
    - CLASSPATH: %JINI_CLASSPATH%;%JINIHOME%\lib\Jama-1.0.1.jar
    - JINI_CLASSPATH: %RUNTIME_JAR%;%JINIHOME%\lib\jini-core.jar;%JINIHOME%\lib\jini-ext.jar;%JINIHOME%\lib\reggie.jar;%JINIHOME%\lib\reggie-dl.jar;%JINIHOME%\lib\tools.jar;%JINIHOME%\lib\gl4java.jar;%JINIHOME%\lib\jARToolkit.jar;%JINIHOME%\lib\gl4java-glffonts.jar;%JINIHOME%\lib\gl4java-glutfonts.jar
    - JINIHOME: .
    - PATH: %JINIHOME%\lib
    - RUNTIME_JAR: <The location of your jdk>\jre\lib\rt.jar
6. Change the directory at the command prompt to the location that the e-DTS 2.0 has been unzipped to. Change to the source (“src”) directory. For example:
    - C:\ → cd C:\DistributedTrackingSystem <Enter>
    - C:\DistributedTrackingSystem\ → cd src <Enter>
7. To compile all of the Java classes simply type “compile” at the command prompt. Once this has completed you are ready to setup the e-DTS 2.0 and begin tracking.
8. In the DistributedTrackingSystem/data/pdf directory you can find all the various patterns that the e-DTS will recognize, simply print off the

desired pattern. The two most common tracking marker patterns are: hiroPatt and patt.kanji.

9. If using the Eclipse IDE you need to specify where the policy file is located. In the Configuration window enter the following in the VM Arguments box:
   - -Djava.security.policy=policy.all
10. Before running the tracking system you need to calibrate each camera in your system. If you have access to the Matlab Utility please skip method (b) otherwise use method (a):

    - To do this you can find the calibration tool in DistributedTrackingSystem/tools directory, named Calibrate_Camera. This tool utilizes the ARToolkit method of calibration. A detailed explanation can be found here: http://www.hitl.washington.edu/artoolkit/documentation/usercalibration.htm

    - For improved Camera Calibration please use the steps provided in [11] to calibrate your camera. This method utilizes the Matlab utility.

11. Once camera calibration has been completed you are ready to run the system. To start the system you must first initiate a JINI Lookup Service, this can be done by typing at the command prompt: “setup.” This will start the JINI Lookup Service; you are now ready to start the system.

12. Once the JINI Lookup Service is running you can type the following at the command prompt:

    - Java CamServiceMulticast <Name>

        - The “Name” parameter is a user specified unique identifier to assign to your camera.

- Java UFilterMulticast <Name>
    - The “Name” parameter is a user specified unique identifier to assign to the Visibility Filter.
- Java MulticastTracker

13. At this point the MulticastTracker will be running and notifying the user of any patterns being seen by cameras participating within the e-DTS 2.0. The MulticastTracker will prompt the user to select either simple averaging fusion or Kalman-based fusion techniques, once a selection has been made the system will start.

Happy Tracking!

Appendix B. Figures

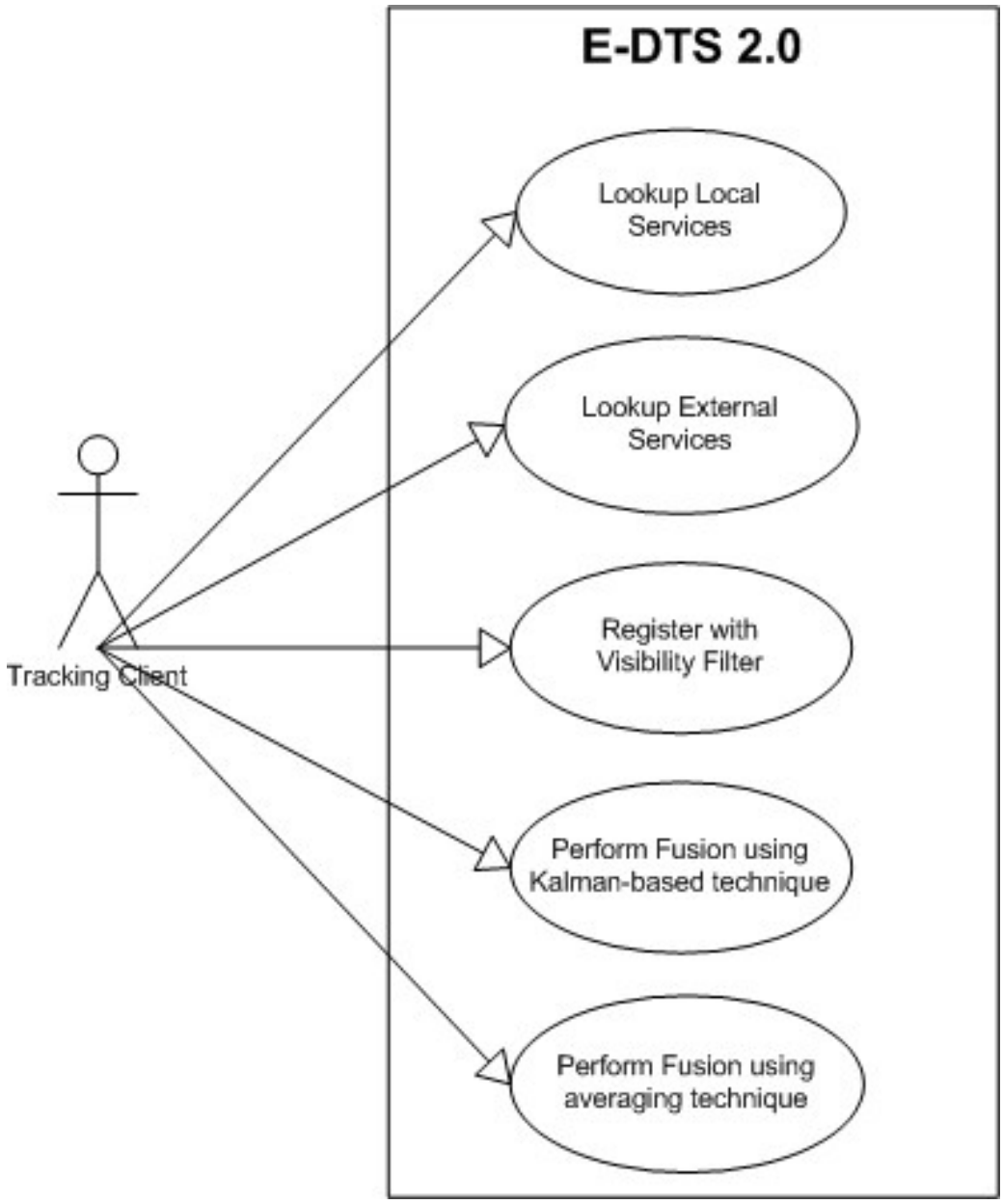

Figure B-1 e-DTS 2.0 Use Case

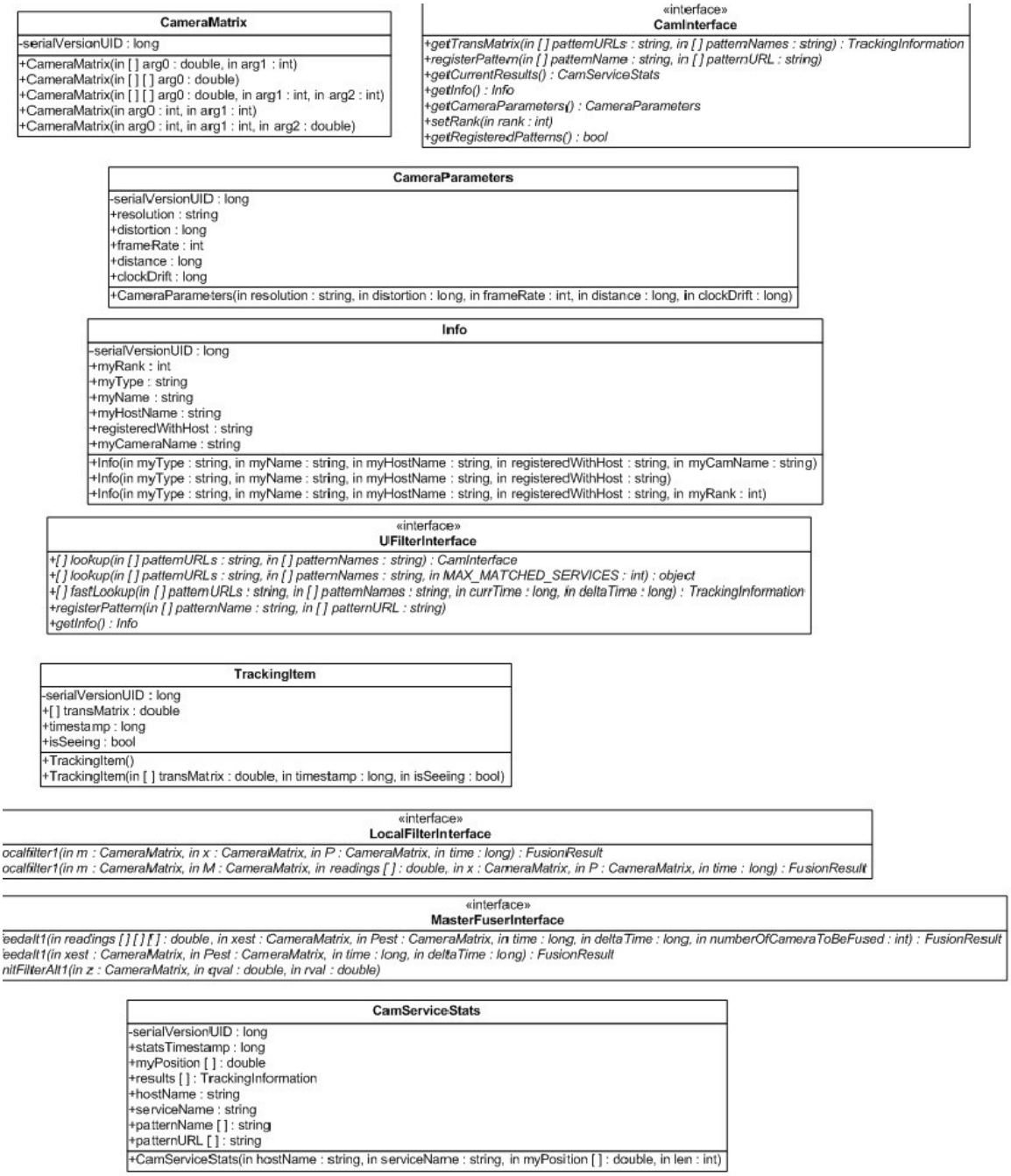

Figure B-2a UML Class Diagram

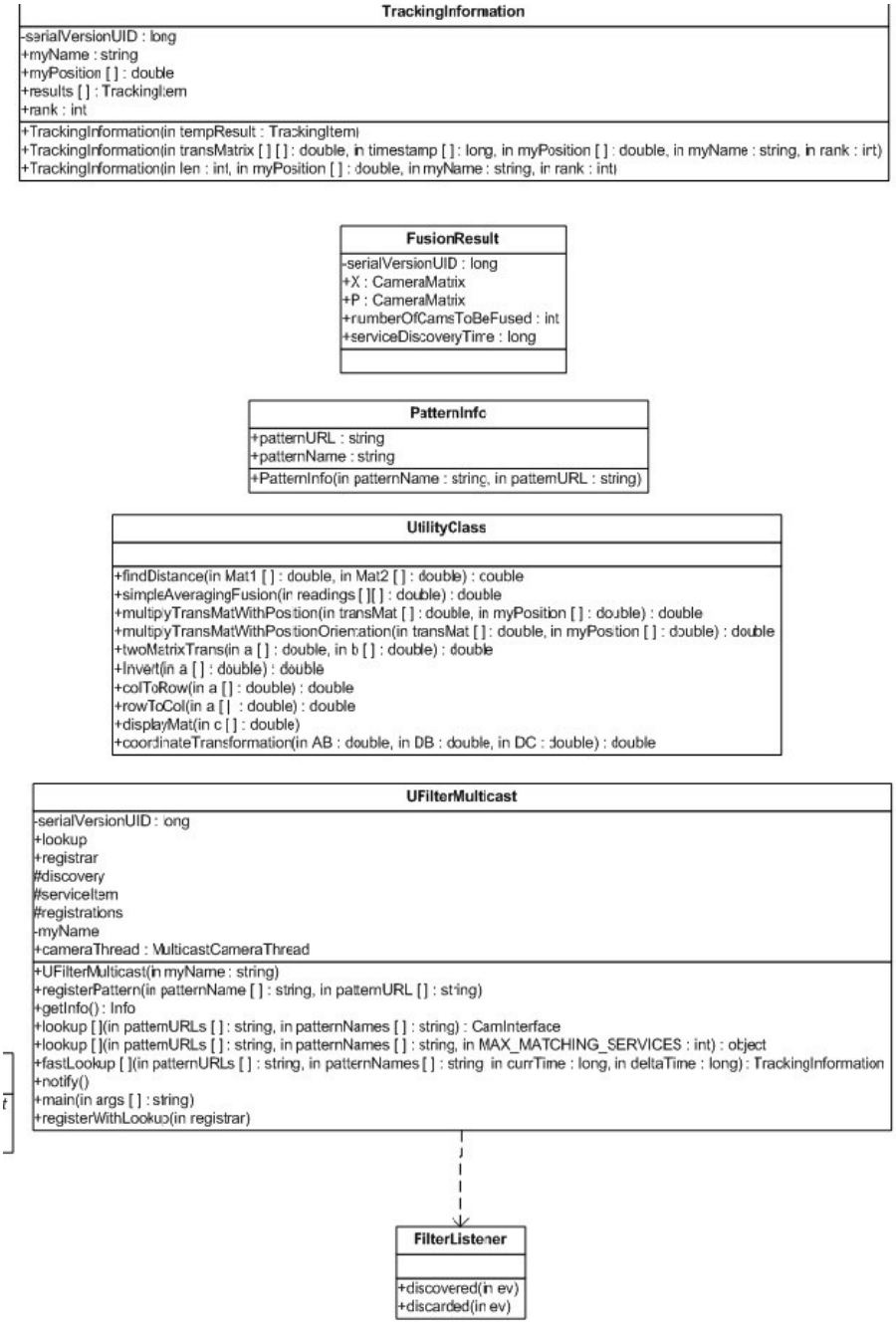

Figure B-3b Class Diagram

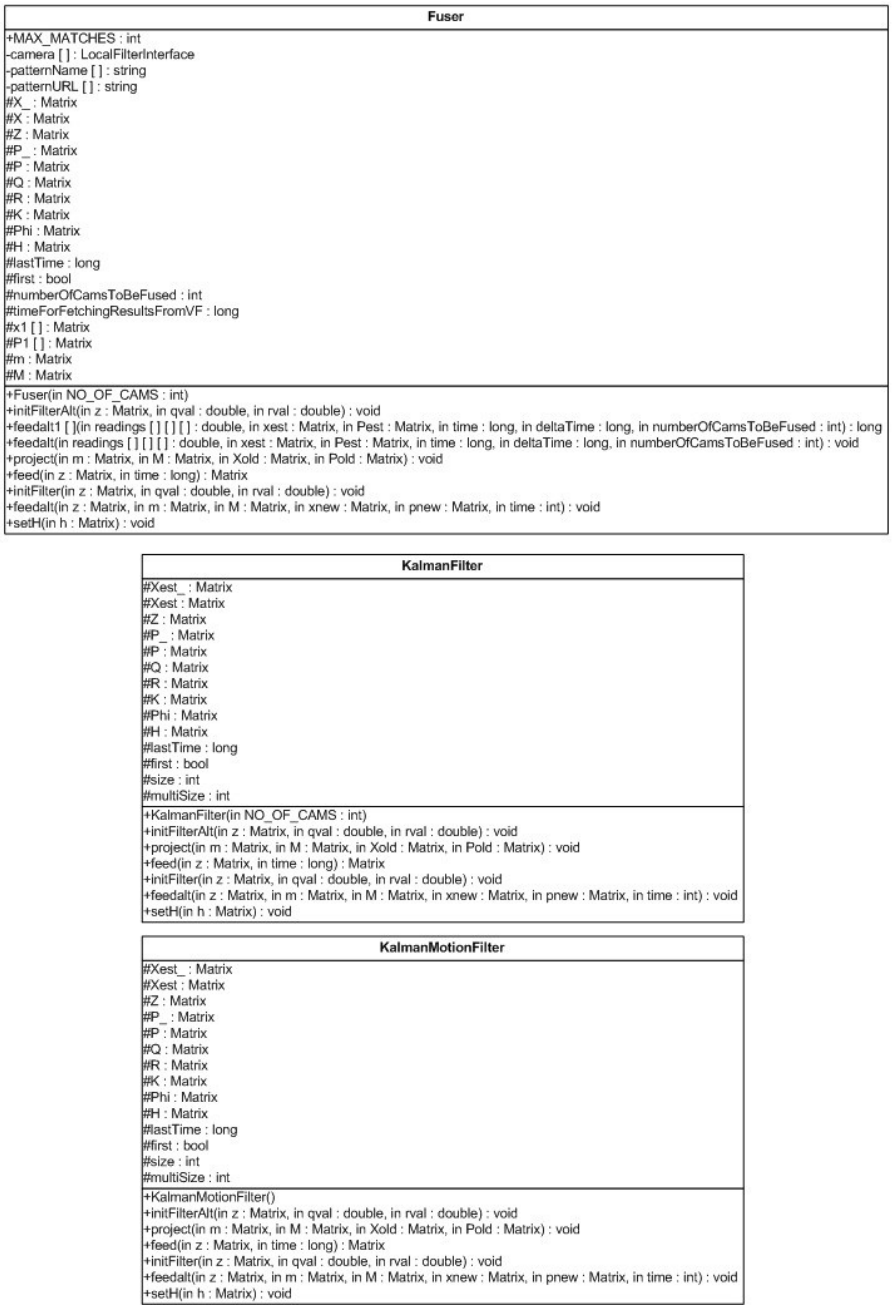

Figure B-4c Class Diagram

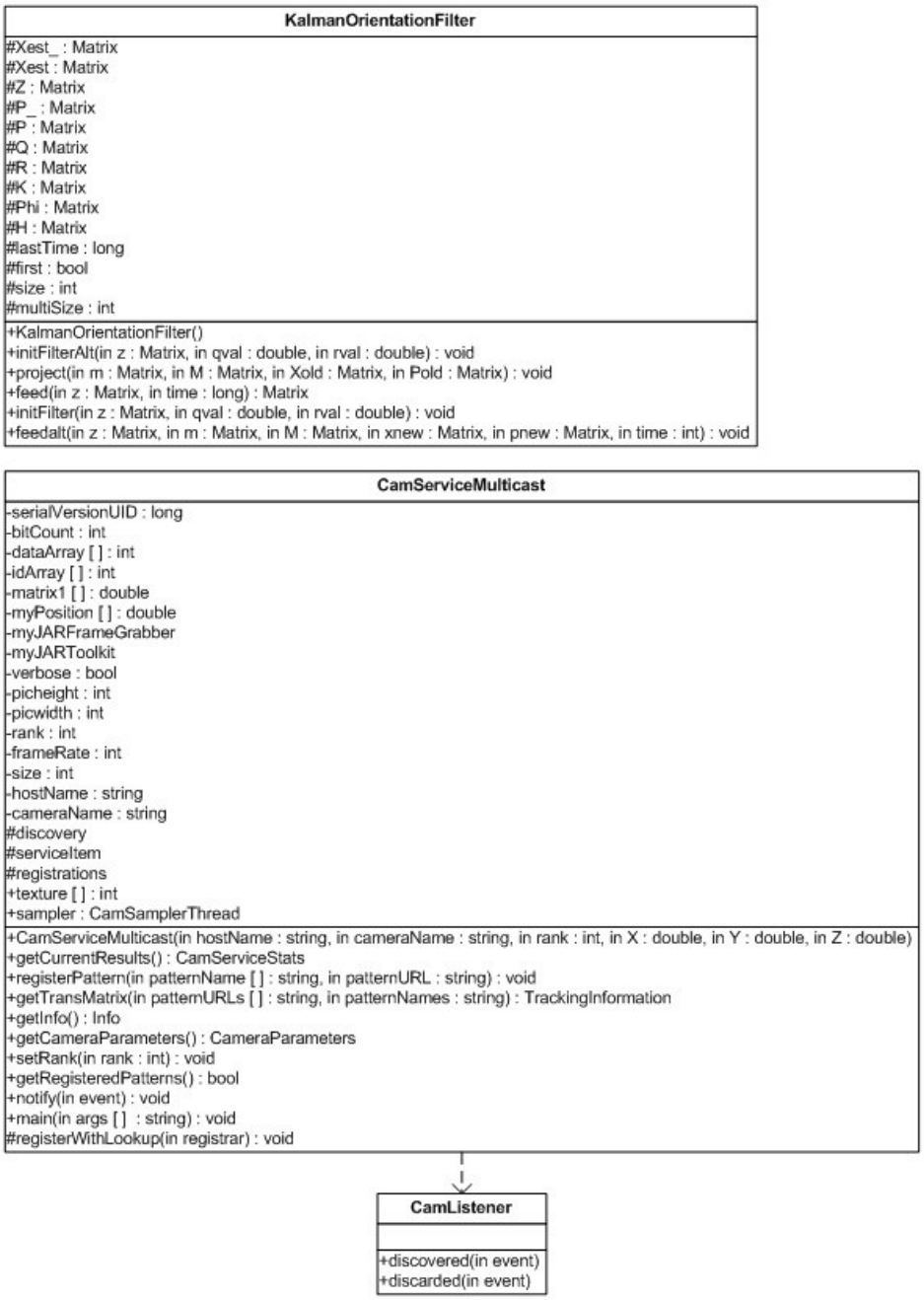

Figure B-5d Class Diagram

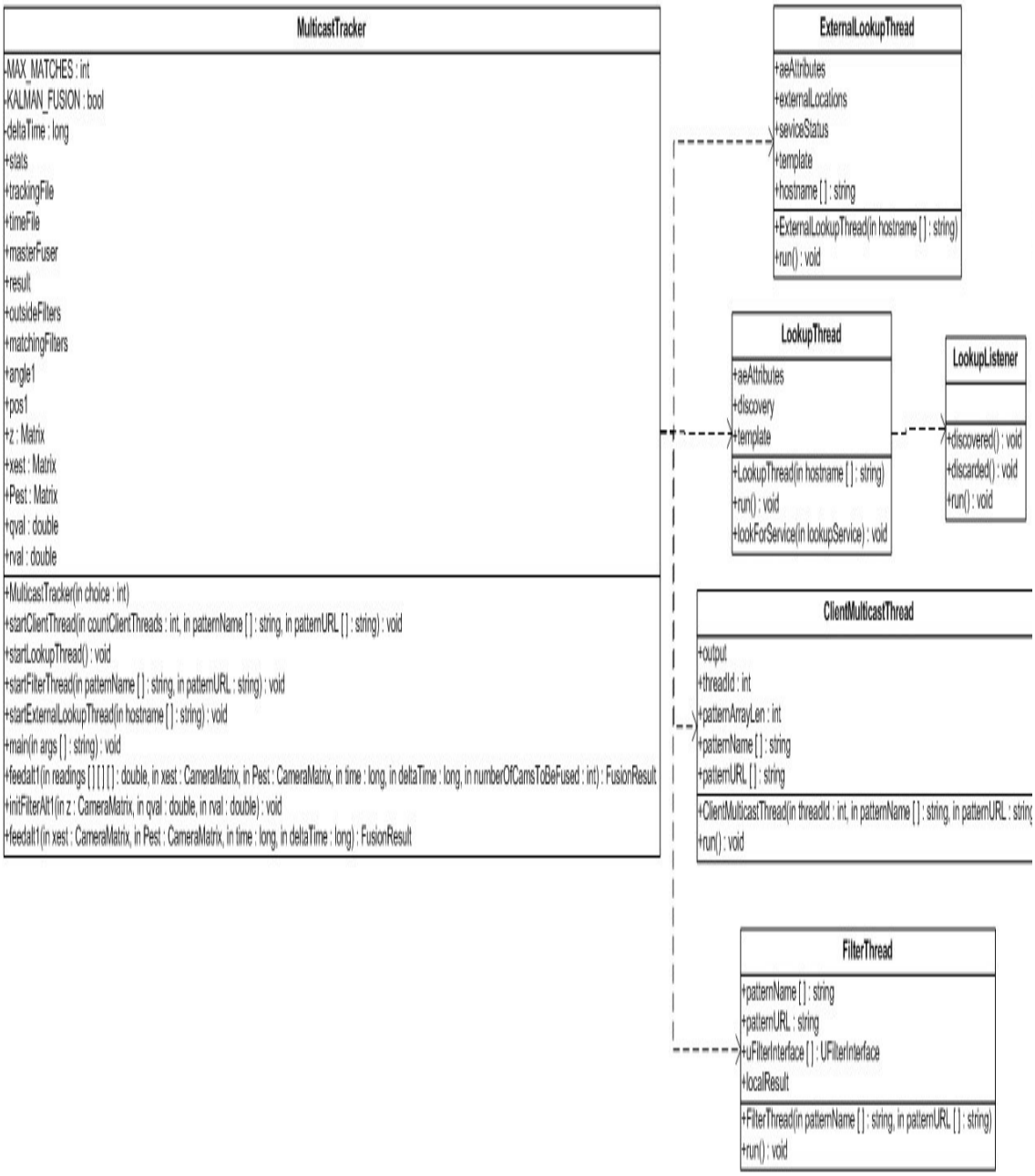

Figure B-6e Class Diagram

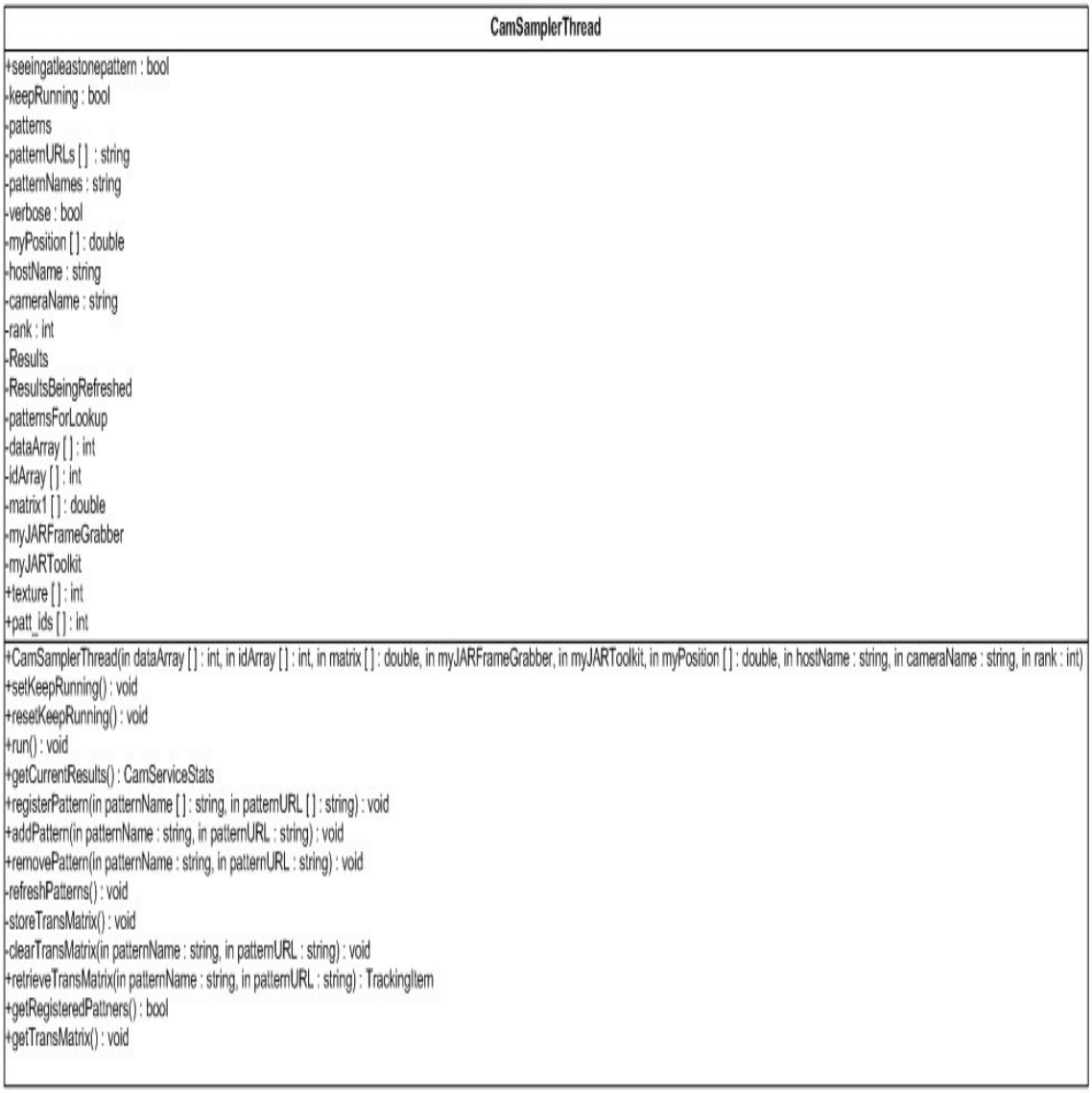

Figure B-7f Class Diagram

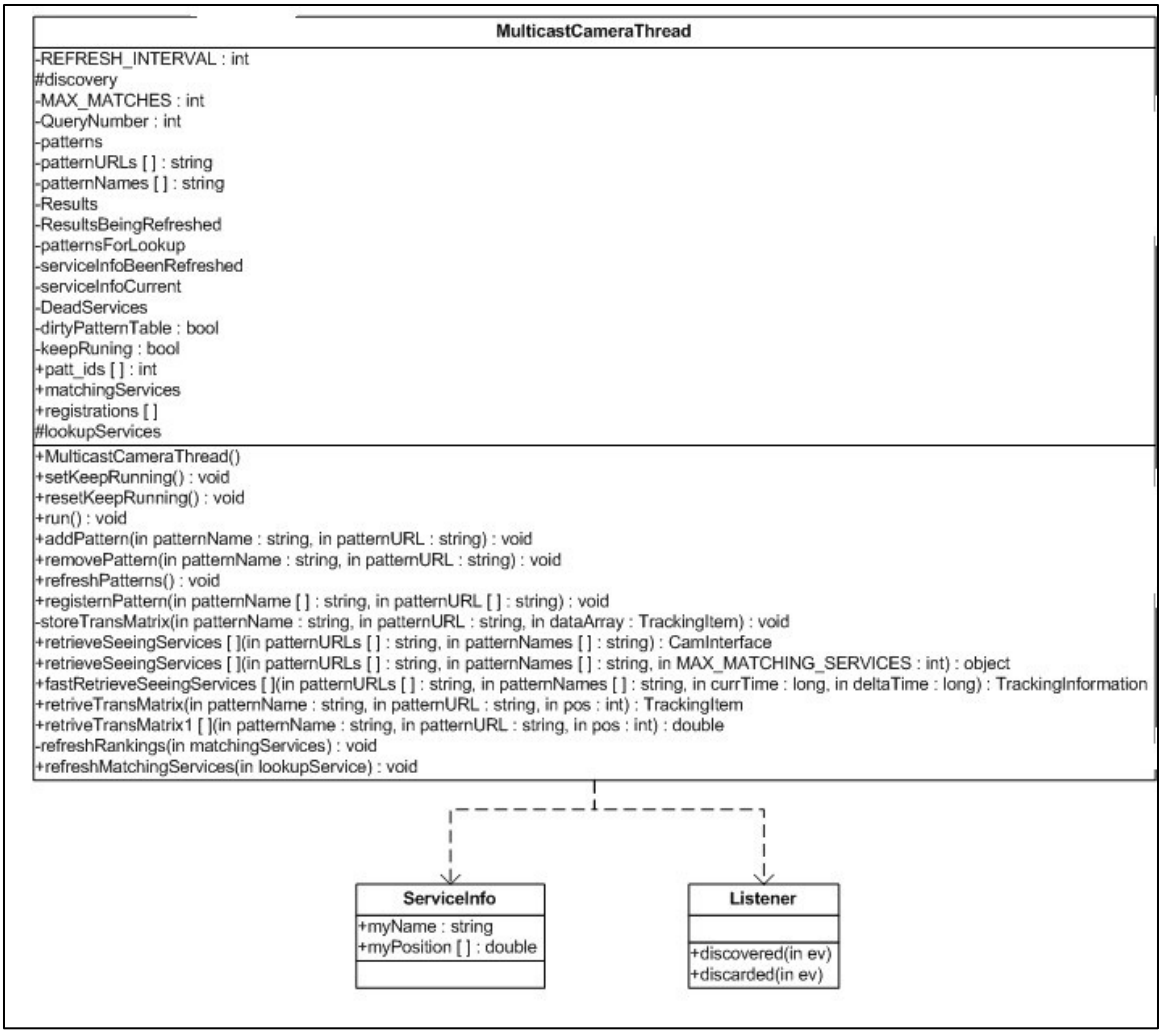

Figure B-8g Class Diagram

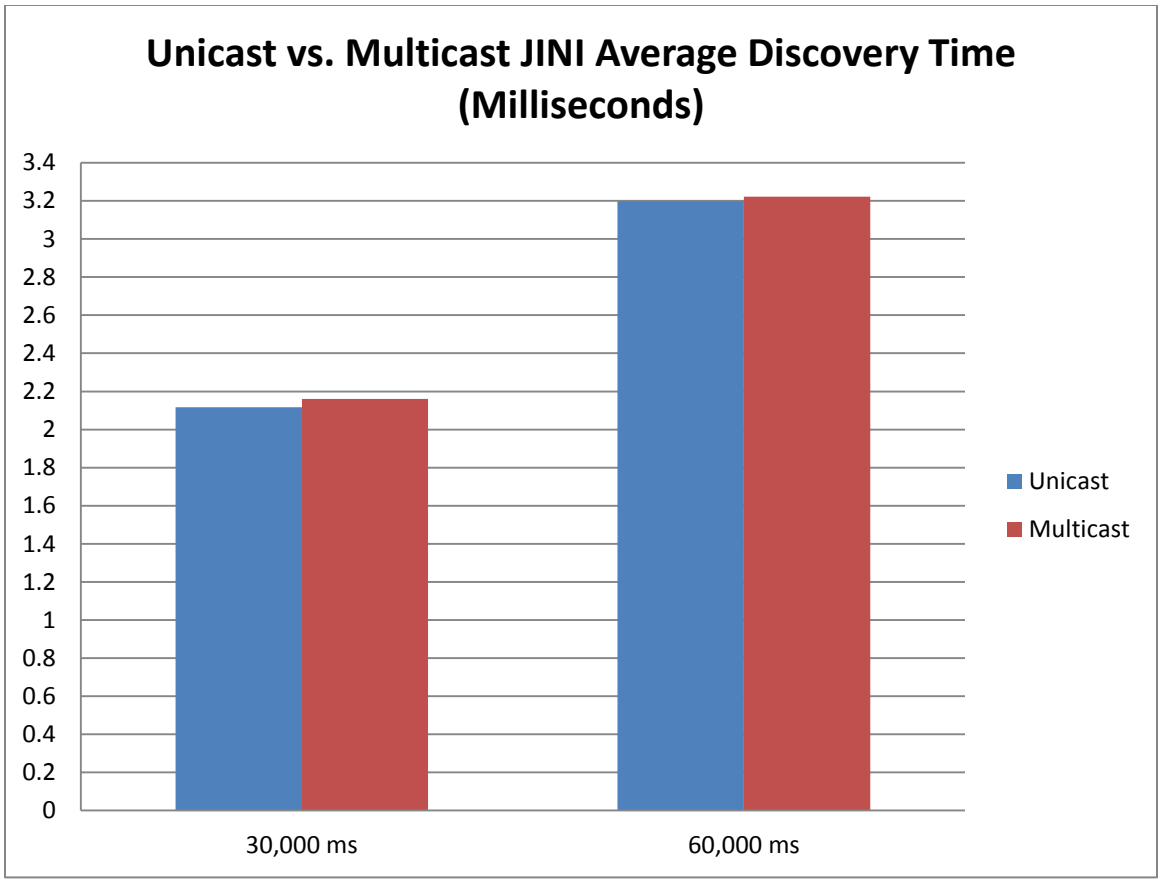

Figure B-9 Average EEDT Chart

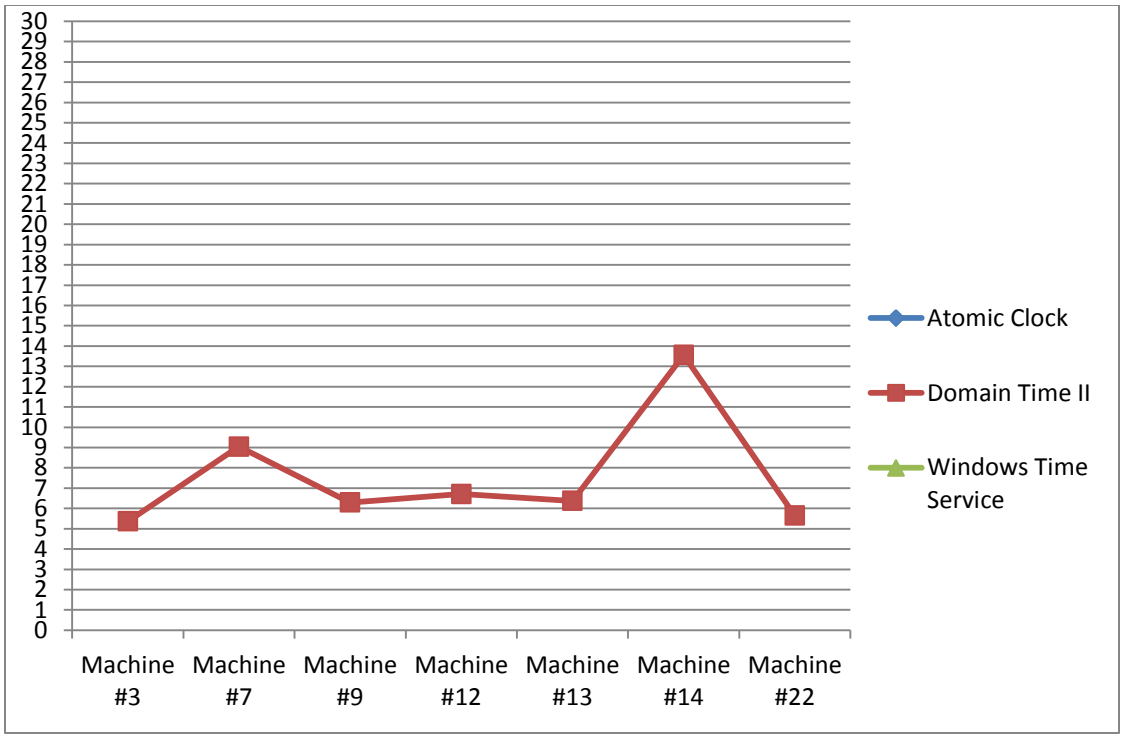

Figure B-10 Clock Synchronization Tool Comparison Graph

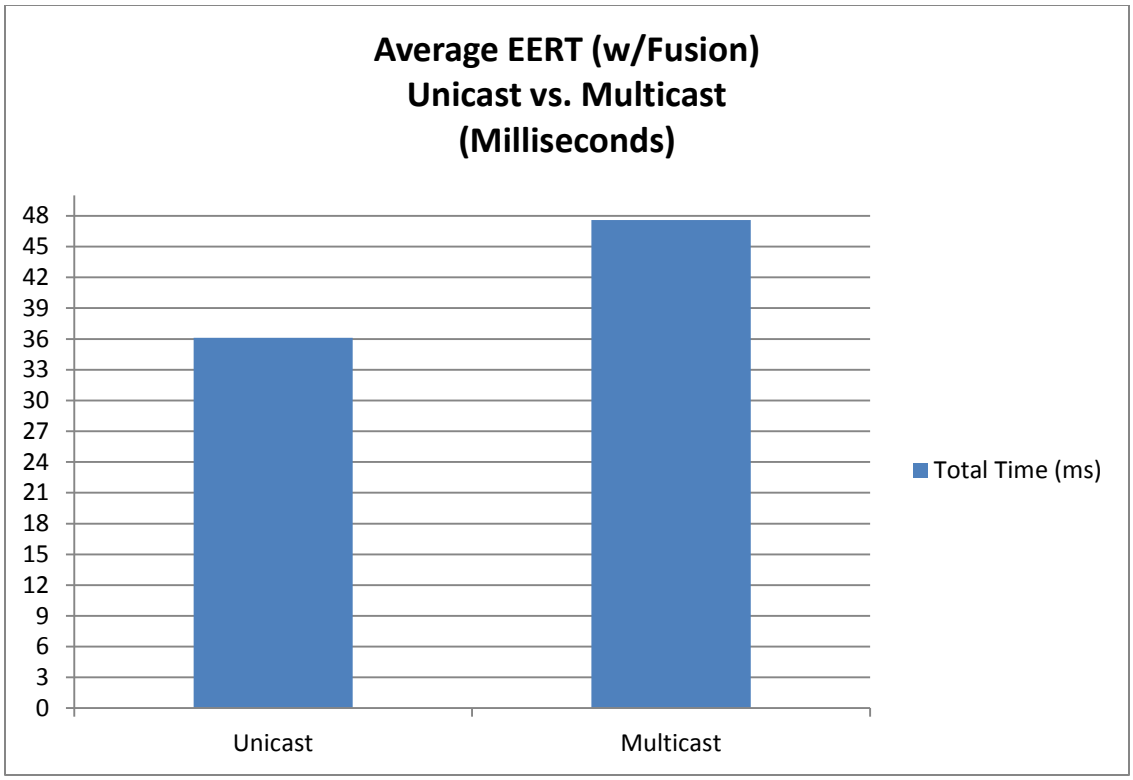

Figure B-11 Average EERT (Unicast vs. Multicast) Chart

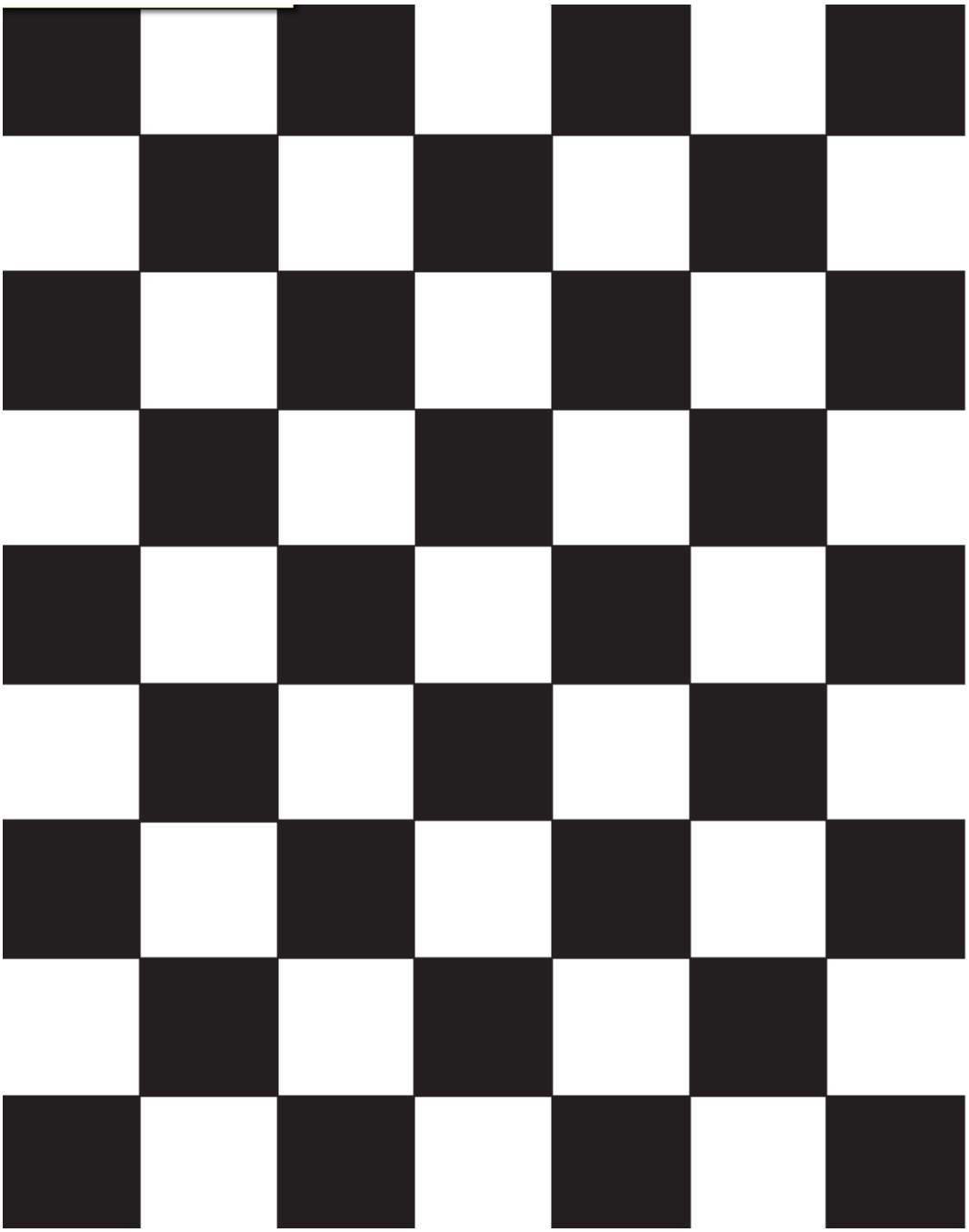

Figure B-12 Calibration Pattern